On the Performance of Cuckoo Search and Bat Algorithms Based Instance Selection Techniques for SVM Speed Optimization with Application to e-Fraud Detection

Andronicus Ayobami AKINYELU[1] and Aderemi Oluyinka ADEWUMI [1]
[1]School of Mathematics, Statistics & Computer Science
University of KwaZulu-Natal, Private Bag X54001 Durban, South Africa 4000
[e-mail: akinyelu.ayobami@gmail.com, adewumia@ukzn.ac.za]
* Corresponding author: Adewumi Oluyinka

Received November 11, 2016; revised May 13, 2017; accepted June 11, 2017; published March 31, 2018

Abstract

Support Vector Machine (SVM) is a well-known machine learning classification algorithm, which has been widely applied to many data mining problems, with good accuracy. However, SVM classification speed decreases with increase in dataset size. Some applications, like video surveillance and intrusion detection, requires a classifier to be trained very quickly, and on large datasets. Hence, this paper introduces two filter-based instance selection techniques for optimizing SVM training speed. Fast classification is often achieved at the expense of classification accuracy, and some applications, such as phishing and spam email classifiers, are very sensitive to slight drop in classification accuracy. Hence, this paper also introduces two wrapper-based instance selection techniques for improving SVM predictive accuracy and training speed. The wrapper and filter based techniques are inspired by Cuckoo Search Algorithm and Bat Algorithm. The proposed techniques are validated on three popular e-fraud types: credit card fraud, spam email and phishing email. In addition, the proposed techniques are validated on 20 other datasets provided by UCI data repository. Moreover, statistical analysis is performed and experimental results reveals that the filter-based and wrapper-based techniques significantly improved SVM classification speed. Also, results reveal that the wrapper-based techniques improved SVM predictive accuracy in most cases.

Keywords: Support Vector Machines, classification, machine learning, phishing email, spam email

1. Introduction

Support Vector Machine (SVM) is a supervised machine learning (ML) algorithm, developed in 1995, for binary classification problems [1]. SVM has been applied successfully to wide range of problems, including pattern recognition [2], email classification [3, 4] and image processing [5]. However, SVM training speed decreases, with increase in dataset size. Its training time is approximately $O(n^2)$, where n, refers to training dataset size [6]. Many SVM speed optimization techniques have been proposed in literature, and most of these techniques tackled optimization from different approaches, including: instance selection, parameter optimization and feature selection. Among these three approaches, instance selection is one of the most efficient [7, 8]. Instance selection helps in increasing classification speed, decreasing memory consumption and improving generalization performance of a classifier. Some instance selection techniques have been proposed in literature, and majority of them are based on k-NN classifier [7]. Also, some techniques are based on k-d trees [9], clustering [10, 11], tabu search [12] and sequential search [13]. However, very few techniques explored Nature Inspired (NI) Algorithms. Some of the few existing NI-based instance selection techniques focused on: Evolutionary Algorithm (EA) [14, 15], Memetic Algorithm [16], Ant Colony Optimization (ACO) [17] and Artificial Immune System (AIS) [18]. This paper propose two wrapper-based and filter-based nature inspired instance selection techniques for improving SVM classification speed and accuracy.

Some applications, such as video surveillance and intrusion detection, requires a classifier to be trained very quickly to enable the classifier identify new target concepts [6]. Moreover, this applications requires the classifier to be trained on large datasets. For this kind of applications, SVM training time can be unacceptably high, which renders SVM ineffectual [6]. Furthermore, even in applications when training can be performed offline (such as spam email filters), if the size of training data or number of classes is large, then SVM computational complexity will be too high [6]. Hence, this paper propose two filter-based instance selection techniques for improving SVM training speed. Fast classification is often achieved at the expense of classification accuracy, and some applications, such as phishing and spam email classifiers, are very sensitive to slight drop in classification accuracy. Hence, the paper also introduces two wrapper-based instance selection techniques, for improving SVM predictive accuracy. The proposed techniques are not limited to SVM, they can also be applied to other ML algorithms.

1.1 E-Fraud Detection

Credit card fraud, phishing and spam email are three prominent e-fraud types that has caused great damages to the global economy in recent times. Spam email refers to unsolicited bulk email [19], mostly sent by individuals trying to advertise products. Phishing refers to unsolicited emails, sent by individuals trying to obtain delicate information from users, usually for the purpose of fraud. Credit card fraud is a term used for fraudulent activities involving credit or debit cards. These three e-fraud types, has caused colossal loss of

millions of dollars. Between October 1st, 2013 and December 1st, 2014, some companies lost a total of $179 million US dollars to email scam. Also, seven thousands companies in USA alone, lost approximately $750 US dollars to phishing, in August 2015 [20]. In 2017, card fraud worldwide is expected to total $27.69 billion US dollars [21]. Unfortunately, e-fraud is on the increase, and fraudsters are devising new sophisticated techniques capable of bypassing existing e-fraud detection systems. Hence, robust e-fraud detection techniques are highly required.

The remaining part of this paper is structured as follows. Section 2 provides a brief introduction to instance selection and a brief survey of existing instance selection techniques. Section 2 also introduces NI techniques and provide a brief survey of existing NI-based speed optimization techniques. Furthermore, Section 3 provide details on the proposed techniques and their experimental results. Finally, the paper is concluded in Section 4.

2. Literature Review

A sizable number of NI-based SVM optimization techniques has been proposed in literature. Most of the proposed techniques focused on feature selection and parameter optimization. Few studies focused on instance selection. This section present a brief survey of some existing instance selection, BA and CSA based techniques.

2.1 Instance Selection

A dataset consist of a collection of redundant (superfluous or harmful) and relevant instances. Superfluous instances refers to instances that contributes negligibly to the decision surface of a classifier, and harmful instances are instances that leads to high false classifications [22]. Instance selection aims to remove superfluous or harmful instances from a dataset. It is a very important preprocessing step in data mining [23, 24], and it can be applied to reduce memory consumption, increase processing speed [25, 26] and improved performance [23]. Instance selection algorithms are divided into two: wrapper and filter [7]. Wrapper-based and filter-based techniques differs in their selection criterion. The selection criterion of wrapper-based techniques depends on the predictive accuracy produced by a classifier, while the selection criteria of filter-based techniques depends on a function, which is independent of a classifier [7]. Chen et al. [27] proposed a filter-based instance selection technique for selecting boundary instances. In the study, firstly, clustering algorithm was used to select cluster centers of positive class instances.

Furthermore, the selected cluster centers were used as references for selecting boundary instances. Authors designed the algorithm on two postulations. Firstly, negative instances near cluster centers of a positive class are close to the boundary, and secondly, positive instances far away from cluster centers of a positive class are close to the boundary. This implies that, positive instances close to a boundary and negative instances far away from a boundary contributes less to the decision surface. Authors performed some experiments to test the efficacy of the proposed technique, and the technique performed well.

In a different work, Hansheng and Venu [28] proposed a new method for improving the computational speed of SVM. In the proposed method, two techniques were combined: Principal Component Analysis (PCA) and Recursive Feature Elimination (RFE). PCA was

used to reduce the dataset dimension, and RFE was used to select relevant features, which in turn, reduced the number of redundant and non-discriminative features. The proposed technique was tested, and it improved SVM computational speed. Additionally, Panda et al. [6] proposed a boundary detection algorithm for improving the speed of SVM. The algorithm was designed to eliminate non-relevant training data instances, that is, instances that are far from a decision boundary. In the study, Panda et al. [6] designed a function that assigns high weights to instances close to a decision boundary. The algorithm was tested on five datasets, and it produced good reduction rates.

2.2 Nature Inspired Techniques

Nature provides some of the best and well-organized ways of solving problems. NI algorithms are inspired by the intriguing problem solving process of natural systems. They have been used to solve many real world complex problems including: hostel allocation problems [29], graph coloring problems [30], annual crop planning problems [31], and email classification [32]. Some NI technique include: Firefly Algorithm (FFA) [33], Particle Swarm Optimization (PSO) [34], Simulated Annealing [35], Cuckoo Search Algorithm (CSA) [36], and Bat Algorithm (BA) [37]. This study presents two filter-based and wrapper-based instance selection algorithms inspired by CSA and BA. A brief introduction to BA and CSA is presented next.

2.2.1 Bat Algorithm

BA is inspired by the echolocation behavior of bats. Most bats uses echolocation to locate food (or preys), to avoid obstacles and to locate their roost in the dark [37]. Bats emits loud sounds in patterns, and pays attention for echo that may reflect back from objects in the surroundings [37]. During hunting, bats emit pulses at a very high rate. However, the rate reduces as they fly closer to a prey [37]. Some bats have good vision, and some have very good smelling ability [37]. This enhances their ability to efficiently detect preys and avoid obstacles [37]. This study propose an instance selection algorithm based on standard BA proposed by Yang [37]. BA was formulated using the following rules [37]:

- All bats use echolocation to detect distance, and they can differentiate between preys and obstacles
- Bats randomly fly, with velocity v_i at position x_i with a fixed frequency f_{min}, varying wavelength λ and loudness A_o to search for preys. Depending on their target proximity, bats can regulate their rate of pulse emission, and the wavelength of their emitted pulses.
- Loudness varies from a large positive value, A_o, to a minimum value, A_{min}.

Pseudocode for BA is given in **Fig. 1**. The position x_i, velocity v_i and frequency f_i for each virtual bats are firstly initialized. Furthermore, they are updated as follows [37]:

$$f_i = f_{min} + (f_{max} - f_{min})\,\beta, \tag{1}$$

$$V_i^t = V_i^{t-1} + \left(X_i^t - X_*\right) f_i, \tag{2}$$

$$X_i^t = X_i^{t-1} + V_i^t \tag{3}$$

where β is a randomly generated number between [0, 1], and X_* is the current global best solution. f_i is used to control speed and range of bat movements. Initially, each bat is assigned a frequency, randomly selected from $[f_{min}, f_{max}]$. Furthermore, new solutions are generated and a solution is selected from the current best set of solutions [37]. Afterwards, a new solution is locally generated for each virtual bat in the population, using random walks:

$$X_{new} = X_{old} + \in A^t, \tag{4}$$

where $\in$ is a random number generated between [-1, 1], and A^t is the loudness of all the bats at every time interval. Additionally, per iteration, the loudness and pulse rate emission are regulated as follows:

$$A_i^{t+1} = \propto A_i^t, \tag{5}$$

$$r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \tag{6}$$

where $\propto and\ \gamma$ are BA parameters. The original BA was proposed for continuous problems. Each virtual bat move in continuous space. However, in instance selection, each bat move in a binary search space, where 1 indicate that an instance is selected and 0 indicate otherwise. In this study, sigmoid function, shown in equation (7), is used to convert each bat positions to binary value.

$$S(V_i^t) = \frac{1}{1+e^{-V_i^t}}, \tag{7}$$

Hence, in place of equation (3), the position of each bat is updated by equation (8):

$$X_i^t = \begin{cases} 1\ if\ \sigma \leq S(V_i^t), \\ 0\quad otherwise, \end{cases} \tag{8}$$

where σ is a random number uniformly drawn from the range [0, 1].

Some BA-based techniques has been proposed in literature. Rodrigues et al. [38] proposed a feature selection approach based on BA and Optimum-Path Forest (OPF). Rodrigues et al. [38] used BA for feature selection, and OPF for classification. The technique was tested and it yielded promising results. In another study, Medjahed et al. [39] used binary CSA to solve the problem of band selection in hyperspectral image classification. In the study, Binary CSA was used to select relevant band subset from dataset. The selected features were then used to train K nearest neighbor (KNN) classifier. The proposed technique produced improved hyperspectral image classification.

Taha et al. [40] proposed a feature selection approach based on BA and Naïve bayes (NB) classifier. Authors used BA for feature selection and NB for classification. The hybridized approach was tested on twelve datasets, and it yielded promising results. Emary et al. [41] combined BA and rough set theory (RST) to solve feature selection problem. In the study, BA was used to extract relevant features from a feature space. Also, authors used RST to

design a fitness function, which considered both classification accuracy and feature size. Authors evaluated the approach and compared it to two other RST-based techniques, and the proposed approach outperformed both techniques. Laamari and Kamel [42] proposed a hybrid technique for intrusion detection, based on BA and SVM. In the study, authors used BA in combination with SVM to solve the problem of intrusion detection. Authors used BA for feature selection and parameter optimization. The hybrid technique was compared to PSO-SVM and standard SVM, and it outperformed both techniques.

2.2.2 Cuckoo Search Algorithm

CSA, proposed by Yang [36], is inspired by the parasitic behavior of some species of cuckoo birds, and the levy flight behavior of some fruit flies and birds species. Some species rely on other birds for hatching their eggs and feeding their young. These species (called brood parasites) lay their eggs in nests of other birds [36]. Mostly, they target nests of birds that newly laid their eggs. Generally, cuckoo eggs hatches earlier than their host eggs, hence, by instinct, the newly hatched cuckoo throws the host eggs out of its nest, to increase the share of food provided by the host bird [36]. CSA was developed based on this parasitic behavior of cuckoos. The following idealized rules were used to develop CSA:

- Each cuckoo lays one egg per time, and randomly distribute its egg to different nest
- The best nest, containing high quality eggs, will survive to the next generation
- Number of host nests is fixed. Also eggs laid by a cuckoo is discovered by the host bird by a probability of $p_a \in [0,1]$. If eggs is discovered, host bird can either abandon its nest and build a new nest, or throw the discovered eggs away.

Pseudocode for CSA is given in **Fig. 2**. In the algorithm, new positions for each cuckoo are generated by performing a levy flight, given in equation (9).

$$X_i^{(t-1)} = X_i^{(t)} + \alpha \oplus Levy(\lambda), \tag{9}$$

where $\alpha > 0$ refers to step size, and it is related to the scales of problem solved. $\oplus$ refers to entrywise multiplication. Levy flight provides random walks, drawn from a levy distribution given in equation (10). The levy distribution has an infinite variance and infinite mean.

$$Levy \sim u = t^{-\lambda}, \qquad (1 < \lambda \leq 3) \tag{10}$$

CSA was originally designed for continuous problem. However, in this study, sigmoid function (shown in equation (11)) is used to convert each cuckoo positions to a binary value (0 or 1). One indicate that an instance is selected, and zero indicate otherwise.

Bat Algorithm

Objective function $f(x), x = (x_1, \dots, x_d)^T$

Initialize Bat population $x_i (i = 1, 2, \dots, n)$ and v_i

Define pulse frequency f_i at x_i

Initialize pulse rates r_i and the loudness A_i

1. While $t < Max\ number\ of\ iterations$
   1.1. Generate new solutions by adjusting frequency and updating velocities and solutions
   1.2. If $(rand < r_i)$
      1.2.1. Select a solution among the best solutions
      1.2.2. Generate a local solution around the selected best solution
   1.3. End if
   1.4. Generate a new solution by flying randomly
   1.5. If $(rand < A_i\ and\ f(x_i) < f(x_*))$
      1.5.1. Accept the new solution
      1.5.2. Increase r_i and $reduce$ A_i
   1.6. End if
   1.7. Rank the bats and find the current best x_*
2. End while
3. Post process result and visualization

Fig. 1. Pseudocode for Bat Algorithm [37].

$$S(V_i^t) = \frac{1}{1 + e^{-V_i^t}}, \tag{11}$$

Hence, in place of equation (9), the position of each cuckoo is updated by equation (12):

$$X_i^t = \begin{cases} 1\ if \sigma \leq S(V_i^t), \\ 0\quad otherwise, \end{cases} \tag{12}$$

where σ is a random number uniformly drawn from the range [0, 1]. CSA has been used in literature, to solve different problems. For example, Rajalaxmia [43] solved the problem of feature selection in Type-2 diabetics using binary CSA and genetic algorithm (GA). In the study, initially, clustering was used for instance selection, afterwards, CSA and GA was used for feature selection. Finally, the selected instances and features were used to build a model for multilayer perceptron (MLP) classifier. In another study, Mousavirad and Ebrahimpour-Komleh [44] proposed a CSA-based technique for feature selection. In the study, authors used CSA for feature extraction. Afterwards, the extracted features were encoded in a binary strings and used to train a k-NN classifier. The proposed approach was evaluated on five datasets obtained from UCI data repository [45], and it yielded good result.

Cuckoo Algorithm via Levy Flight

Objective function $f(x), x = (x_1, \dots, x_d)^T$

Generate initial population of n host nests x_i $(i = 1, 2, \dots, n)$

1. While $(t < MaxGeneration) or (stop\ crieterion)$
   1.1. Get a cuckoo randomly Levy flights
   1.2. Evaluate its quality or fitness F_i
   1.3. Choose a nest among n (say, j) randomly
   1.4. If $(F_i > F_j)$
      1.4.1. Replace j by new solution;
   1.5. End if
   1.6. A fraction (P_a) of worse nests are abandoned and new ones are built
   1.7. Keep the best solutions (or nest with quality solutions);
   1.8. Rank the solutions and find the current best
   1.9. Rank the bats and find the current best x_*
2. End while
3. Post process result and visualization

Fig. 2. Pseudocode for Standard Cuckoo Search Algorithm [36]

3. Proposed Algorithms

This study presents two filter-based and wrapper-based instance selection algorithms for optimizing SVM training speed and predictive accuracy. The first technique is called Cuckoo Search Instance Selection Algorithm (CSISA), and the second technique is called Bat Instance Selection Algorithm (BISA). The main difference between the filter-based and wrapper-based techniques is in their objectives. The primary objective of the filter-based techniques is to improve SVM training speed, and the primary objective of the wrapper-based techniques is to improve SVM predictive accuracy. **Fig. 3** shows the pseudocode for BISA, **Fig. 4** shows the pseudocode for CSISA and **Fig. 5** shows the flowchart for the two algorithms.

3.1 Proposed Cuckoo Instance Selection Algorithm

The algorithm starts by initializing the positions for each nest and other parameters, including Min, where Min is the minimum number of instances to be selected for training. Each nest position is initialized to 0 or 1, where 1 indicates that an instance is selected and 0 indicate that an instance is not selected. Furthermore, the initialized solutions are evaluated and the current best solution is kept. Afterwards, new solutions are constructed by randomly selecting different cuckoos through levy flight. The value of each new solution is continuous, hence, they are converted back to binary values after construction. Furthermore, the quality (or fitness) of each nest is evaluated and the global best solution is saved. Fitness function for the filter and wrapper based CSISA are reported in sections 3.1.1 and 3.1.2 respectively.

This process is repeated until a user-defined threshold is reached. Afterwards, the global best cuckoo is selected and N instances are extracted from it, where N is the total number of instances selected by the global best cuckoo. Moreover, N is compared to Min, and if N is less than Min, then Q additional instances are randomly selected from the training dataset and added to the global best cuckoo, where $Q = N - Min$. Finally, the global best cuckoo is selected, and instances with the value of 1 are extracted and used to train SVM.

3.2 Proposed Bat Instance Selection Algorithm

BISA is inspired by the echolocation of bats. It begins by defining the pulse rate and loudness for each artificial bat and also initializing each bat solution to a binary value, where 1 indicate that an instance is selected and 0 indicate otherwise. Furthermore, fitness value for each solution is calculated and the best solution is kept. Fitness function for the filter and wrapper based BISA is reported in section 3.1.1 and 3.1.2 respectively. New solutions are constructed by constructing new frequency and velocity for each bat using equation (1) and equation (2) respectively. Afterwards, each solution is evaluated and the global best solution is updated if a better solution is found. Moreover, a random number ($rand$) is generated and new solutions are constructed if $rand$ is greater than a user-defined pulse rate. The new solutions are retained if $rand$ is less than a user-defined bat loudness (A_i). This process is repeated until a user-defined threshold is reached. Afterwards, the global best bat is selected and N instances are extracted from it, where N is the total number of instances selected by the global best bat. Moreover, N is compared to Min, where Min is the minimum number required training instances. If N is less than Min, then Q additional instances are randomly selected from the training dataset and added to the global best agent, where $Q = N - Min$. Finally, the global best bat is selected, and instances with the value of 1 are extracted and used to train SVM. Pseudocode for the algorithm is shown in **Fig. 4**.

Cuckoo Search Instance Selection Algorithm

Notation

D: Dataset

NF: Number of Folds for Cross Validation

NI: Number of Iterations

NS: Number of Selected Instances

Min: Minimum number of selected instances

G(x): Fitness Function

IM: Instance mask

CB: Current Best

TS: Training Subset

CA: Classification Accuracy

ACA: Average Classification Accuracy

MaxG: Maximum Generation

N: Population Size

NF: Number of Folds for SVM Cross Validation

GB: Global Best

CA: Classifier Accuracy

FT: User-defined Fitness Threshold

Input: NF, NI, MaxG, N, Min, D, FT

Output: ACA

1. **Start** CSISA
2. **For** i = 1 to NF
   - 2.1. Select subset (i.e. 9/10 of dataset) for training
   - 2.2. Pass training subset to CISA for instance selection
   - 2.3. **Start** CISA
     - 2.3.1. Define G(x) for cuckoo nests
     - 2.3.2. Initialize Parameters
     - 2.3.3. **For** a = 1 to N
       - 2.3.3.1. Initialize solution for $nest_a$
     - 2.3.4. **End** for
     - 2.3.5. Evaluate G(x) and select CB
     - 2.3.6. GB = CB
       - 2.3.6.1. **While** (j < MaxG)
         - 2.3.6.1.1. **For** k = 1to N
           - 2.3.6.1.1.1. Construct new solutions by randomly selecting cuckoos using levy flight
           - 2.3.6.1.1.2. Convert new solutions to binary
         - 2.3.6.1.2. **End** k
         - 2.3.6.1.3. **For** a = 1 to N
           - 2.3.6.1.3.1. Replace low quality nest by generating new solutions. Low quality nests are discovered with a defined probability
           - 2.3.6.1.3.2. Convert new solutions to binary
         - 2.3.6.1.4. **End** a
         - 2.3.6.1.5. **For** a = 1 to N
           - 2.3.6.1.5.1. Evaluate G(x_a) for new $solution_a$
         - 2.3.6.1.6. **End** a
         - 2.3.6.1.7. **If** GB > FT
           - 2.3.6.1.7.1. **End While**
         - 2.3.6.1.8. **End** if
       - 2.3.6.2. **End** While
       - 2.3.6.3. Get NS from GB
       - 2.3.6.4. **If** NS < Min
         - 2.3.6.4.1. Add (Min - NS) instances to GB
       - 2.3.6.5. **End** if
       - 2.3.6.6. Output GB
   - 2.4. **End** CISA
   - 2.5. Train SVM model on instances selected by GB
   - 2.6. Test model on current test data (i.e. 1/10 of dataset)
   - 2.7. Sum CA

3. **End for**
4. Calculate ACA, over number of folds
5. Output ACA; ACA = CA / NF
6. **End** CISA_SVM

Fig. 3. Pseudocode for CISA

Bat Instance Selection Algorithm

Notation

D: Dataset

NF: Number of Folds for Cross Validation

NI: Number of Iterations

NS: Number of Selected Instances

Min: Minimum number of selected instances

G(x): Fitness Function

IM: Instance mask

CB: Current Best

TS: Training Subset

PR: Pulse Rate

L: Loudness

CA: Classification Accuracy

ACA: Average Classification Accuracy

MaxG: Maximum Generation

N: Population Size

NF: Number of Folds for SVM Cross Validation

GB: Global Best

CA: Classifier Accuracy

FT: User defined Fitness Threshold

Input: NF, NI, MaxG, N, Min, D, FT

Output: ACA

1. **Start** BISA_SVM
2. **For** i = 1 to NF
   - 2.1. Select subset (i.e. 9/10 of dataset) for training
   - 2.2. Pass training subset to BISA for instance selection
   - 2.3. **Start** BISA
     - 2.3.1. Define G(x) for bats

2.3.2. Initialize Parameters
2.3.3. **For** a = 1 to N
2.3.3.1. Initialize solution for bat_a
2.3.3.2. Define pr_a for bat_a
2.3.3.3. Define l_a for bat_a
2.3.4. **End** for
2.3.5. Evaluate G(x) and select CB
2.3.6. GB = CB
2.3.7. **While** (j < MaxG)
2.3.7.1. **For** k = 1to N
2.3.7.1.1. Construct new frequency for bat_k by using
$f_{min} + (f_{max} - f_{min})\,\beta$
2.3.7.1.2. Construct new velocity for bat_k using
$V_k^t = V_k^{t-1} + (X_k^t - X_*)\,f_k$
2.3.7.1.3. Generate Random Number, R
2.3.7.1.4. **If** R > pr_k
2.3.7.1.4.1. Construct a solution around GB
2.3.7.1.5. **End** if
2.3.7.1.6. Convert bat_k to binary
2.3.7.2. **End** k
2.3.7.3. **For** a = 1 to N
2.3.7.3.1.1. Generate Random Number, R
2.3.7.3.1.2. Evaluate G(x_a) for new solution
2.3.7.3.1.3. Replace previous $solution_a$ with new $solution_a$, if new solution it is better, and if if R < l_a
2.3.7.4. **End** a
2.3.7.5. **If** GB > FT
2.3.7.5.1. **End While**
2.3.7.6. **End** if
2.3.8. **End** While
2.3.9. Get NS from GB
2.3.10. **If** NS < Min
2.3.10.1. Add (Min - NS) instances to GB
2.3.11. **End** if
2.3.12. Output GB
2.4. **End** CISA
2.5. Train SVM model on instances selected by GB
2.6. Test model on current test data (i.e. 1/10 of dataset)
2.7. Sum CA
3. **End for**
4. Calculate ACA, over number of folds
5. Output ACA; ACA = CA / NF
6. **End** CISA_SVM

Fig. 4. Pseudocode for BISA

3.1 Fitness Function

As aforementioned, the primary difference between the proposed filter-based and wrapper-based techniques is in their objectives. This section presents the fitness function for the proposed filter-based and wrapper-based techniques.

3.1.1 Fitness Function for the Proposed Filter-based Techniques

Fitness function for the proposed filter-based BISA and CSISA is shown in equation (13). The fitness function considers both percentage reduction and boundary instances. More weight is assigned to agents with high percentage reduction and high number of boundary instances. The fitness function evaluation begins by calculating the total number of instances in each agent (α). Furthermore, the algorithm calculates the number of instances selected by each agent (β) and the number of boundary instances selected by each agent (γ). The number of instances selected by an agent is obtained by adding all the non-zero elements in the instance mask of the agent. Also, the number of boundary instances selected by an agent is obtained, by firstly passing its selected instances to a boundary detection algorithm for boundary instance selection. Furthermore, the algorithm selects boundary instances, and the number of selected boundary instances is calculated and used for fitness value evaluation. In this study, clustering-based boundary detection algorithm, proposed by Chen et al. [27], is used for boundary instance selection. Finally, α, β and γ are used to calculate the fitness value, as shown in equation (13).

$$fitness_i = \left(\left(100 * \frac{\alpha - \beta}{\alpha} \right) + \left(\frac{\gamma}{\beta} * 100 \right) \right) \Big/ 2 \tag{13}$$

where α = total number of instances in an instance mask, β = number of selected instances in an instance mask and γ = number of selected boundary instances.

3.1.2 Fitness Function for the Proposed Wrapper-based Techniques

Fitness function utilized by the wrapper-based instance selection techniques is shown in equation (14). The primary objective of the proposed wrapper-based techniques is to improve the classification accuracy of SVM. Hence, the fitness function is calculated by computing the classification accuracy of the candidate solution constructed by each agent. That is, for each candidate solution, a classification model is constructed by training the constructed solution (i.e. the reduced subset) on a classifier. Afterwards, the constructed model is evaluated by testing it on a new dataset (test dataset), and the resultant classification accuracy is used as the fitness value for the candidate. The candidate with the best fitness value is the candidate with the highest classification accuracy. The best candidate will be selected and used to build the final classifier.

$$fitness_i = \alpha_i \tag{14}$$

where α_i is the classification accuracy for each candidate in the solution space.

3.2 Extracted Features

Prior to classification, some set of spam features are extracted from each email in the spam email datasets used for evaluation. After extraction, the features are formatted according to the input format required by libSVM [46], and saved in a text file for easy processing.

LibSVM is the SVM library used in this research for all experiments. Details on the extracted spam features are described in this section. The features used for phishing email classification is similar to the features used in one of our previous studies [4].

3.2.1 Word-Based Features

For this feature, different words are extracted from all emails in the dataset, using the extraction technique proposed by Paul Graham [47]. Moreover, spam score for each word is calculated, and words with high spam score are selected and used as a feature. In this study, a total of N word-based features are extracted, where N is the number of words with spam score greater than, or equal to 0.9999.

3.2.2 Term Frequency + Inverse Sentence Frequency

This feature is a combination of term frequency (TF) and inverse sentence frequency (ISF). For each email, TF for each word is calculated using equation (15), and ISF for each sentence in an email is calculated using equation (16). Finally, as shown in equation (17), sum of the product of TF and ISF is calculated and used as a feature. In this study, we converted this feature to binary by assigning 0 to emails with TF-ISF value less than 100, and 1 to emails with TF-ISF values greater than 100. This feature was also used by Shams and Mercer [48].

$$TF_t = \begin{cases} 1 + \log(frequency), if\ frequency_t > 0 \\ 0\ Otherwise \end{cases} \tag{15}$$

$$ISF_t = \log \frac{N}{SF_t}, \tag{16}$$

$$where\ N\ is\ message\ length, and\ SF_t\ is\ number\ of\ sentences\ with\ term\ t$$

$$\sum_t TF_t \times ISF_t \tag{17}$$

3.2.3 Complex Words

Words with more than two syllables are called complex words. In this study, emails containing less than fifteen complex words are assigned the value of 0, and emails containing more than fifteen complex words are assigned the value of 1. This feature was proposed by Shams and Mercer [48].

3.2.4 Simple Words

Simple words refers to words with one or two syllables. A Boolean value of 0 is recorded if an email contain less than fifty simple words, and 1 is recorded if an email contain more than fifty simple words. This feature is similar to the feature used by Shams and Mercer [48].

3.2.5 Spam Words

Some list of spam words, provided by Sham and Mercer [48], are extracted and used as features. A Boolean value of 1 is recorded if an email contain more than one spam word, and 0 is recorded otherwise.

3.2.6 Total HTML Tags

HTML tags are keywords that defines how web browsers formats and displays contents [49], such as text and images. HTML tags are extracted from each email and a Boolean value of 1 is recorded if an email contain more than one HTML tag and 0 is recorded otherwise. This feature was also used by authors in [48].

3.2.7 Document Length

Document length refers to the number of sentences in an email document. A Boolean value of 1 is recorded if an email contain more than one sentence, and 0 is recorded otherwise. This feature was proposed by Shams and Mercer [48].

3.2.8 Non Anchor Tags

HTML anchor tags (<a><a/>), are tags used to navigate to other web pages. All tags that are not anchor tags (such as <p> and <h1>), are extracted from each email and a Boolean value is recorded. Emails containing more than one non-anchor tag is assigned the value of 1, and emails containing one or no non-anchor tag is assigned the value of 0. This feature was also used by Shams and Mercer [48].

3.2.9 Stop Words

Stops words are words frequently used in a specific language. Some list of stop words, provided by Shams and Mercer [48], are extracted from each email and a Boolean value is recorded. Emails with stop words greater than hundred, are assigned the value of one, and emails containing less than hundred stop words are assigned the value of zero. This feature was proposed by Shams and Mercer [48].

3.2.10 Presence of 'Link', 'Click Here' in URL Text of a Link

Most spam or phishing email typically requires users to click on a link, which re-directs them to a spam or phishing websites. Hence, for each email, URLs are extracted, and a Boolean value of 1 is recorded based on whether the URL text contains the following words: "Click Here" or "Link". Otherwise, 0 is recorded. Similar feature was used by authors in [4].

3.2.11 Domain Name Disparity

Domain names are used to detect different web pages. For example, the domain name of "https://www.google.com/" is "google.com". Domain names in the body of legitimate emails, should be similar to the sender's domain name. If there is a disparity, the email is likely a spam email. Domain names from the body section of each email are extracted and compared to the domain name used to send the email. If there is a disparity, the email is assigned the value of one, otherwise, the email is assigned the value of zero. This feature was also used in [4] and [50].

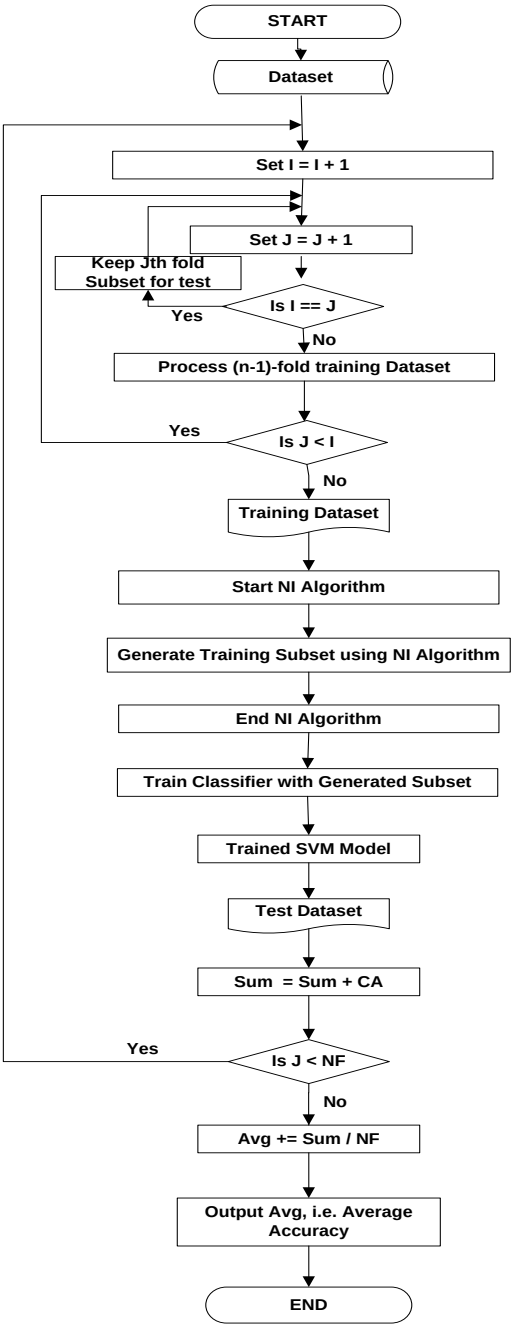

Fig. 5. Flowchart for proposed NI Algorithm

3.2.12 Sum of Distinct Domain

As aforementioned, domain names are used to detect web pages. Domain names are extracted from each email and the total number of domain names is recorded and used a continuous feature. Domain names that appears more than once are counted only once. This feature was also used by authors in [51] and [4].

3.2.13 SpamAssassin Feature

SpamAssassin is a reliable spam email filter, currently used by some organizations. In this study, SpamAssassin is used to classify each email and a Boolean value of 1 or 0 is assigned to an email based on the output of SpamAssassin. An untrained offline version of SpamAssassin is used with the default threshold value and rule weights. Similar feature was used by Akinyelu et al. [4] and Fette et al. [51].

3.2.14 HTML Content Type

Emails are of different formats and content types. These standards and formats are defined by MIME standards. Email content type could be "ordinary text", or "HTML". Ordinary text content type is defined by "text/plain", and "HTML" content type is defined by "text/html". Fette et al. [51], noted that emails with "HTML" content type, are likely scam emails. Hence, in this study, emails with "text/html" are assigned the value of one, otherwise, emails are assigned the value of zero. Similar feature was also used in [4] and [51].

3.2.14 Total Email Links

Zhang and Y. Yuan [52] pointed out that emails containing many URLs are likely spam or phishing emails. Hence, email links are extracted from each email and the total number of links are recorded and used as a continuous feature. This feature was also used by authors in [4] and [52].

3.3 Experimental Setup

The proposed techniques are validated on datasets containing three popular e-fraud types: credit card fraud, phishing email and spam email. The first dataset (Dataset 1) contain 3500 ham email provided by SpamAssassin [53] and 500 phishing emails provided by Jose Nazario [54]. Currently, the phishing emails are no longer available online. Interested users are advised to contact the dataset provider, Jose Nazario [54]. The second dataset (Dataset 2) contain 3500 ham emails and 500 spam emails provided by SpamAssassin [53]. The third dataset (Dataset 3) contains 2787 ham emails and 1813 spam emails, provided by UCI data repository [55]. The fourth dataset (Dataset 4) contains 492 credit card fraud transactions and 4508 legitimate card transactions, provided by Andrea [56]. **Table 3** shows a summary of the four datasets. In addition, robustness of the techniques are further demonstrated by validating them on 20 datasets, provided by the popular UCI dataset repository [55].

The performance of CSISA and CISA is compared to standard SVM and eight existing instance selection techniques in terms of predictive accuracy, classification speed and storage reduction. The compared techniques include: CLUS_IS [27], KNN_IS [6], PSC [57], DROP 3 [26], DROP 5 [26], GCNN [58], POC-NN [59] and ADR-Miner [17]. All experiments are performed using the popular 10 times, 10 fold cross validation technique. Furthermore, sixteen features are extracted from Dataset 1 and fifteen features are extracted from Dataset 2. The features extracted from Dataset 1 is similar to the features described in one of our previous studies [4]. Also, the features extracted from Dataset 2 are described in Section 3.2. After feature extraction, information gain (IG) for all the extracted features was

generated and features with high IG are used for training. For Dataset 1, the best 9 features are used for training, and for Dataset 2, the best 10 features are used for training. Dataset 3, Dataset 4 and the UCI datasets, were already processed by their providers, hence feature extraction was not necessary. All the datasets are formatted according to the input format required by libSVM [46] - the SVM Library used in this study. Moreover, all the features are scaled down using Gaussian Transformation.

Given N training instances, utilizing all the training set for training is time consuming. Instead of utilizing the entire training set, training a classifier on a reduced subset, void of superfluous or harmful instances, will not significantly affect the classification accuracy of the classifier, rather, it can lead to similar or improved classification accuracy [22]. On this basis, the proposed filter-based techniques are designed to use only a subset of the entire training set for instance selection. That is, for all experiments, n instances are passed to BISA and CSISA for processing, where $n < N$. This implies that BISA and CSISA searches an instance space consisting of n instances, instead of an instance space consisting of N instances (i.e. the entire training dataset). Furthermore, for all the experiments, different set of parameters are evaluated, with the aim of determining the best parameters suitable for the proposed techniques and also demonstrating the robustness of the proposed techniques. Results for the best parameters are reported in section 3.4. Unlike the filter-based techniques, the wrapper-based techniques are designed to use the entire training set. That is, they are designed to search through the entire training data for relevant instances.

For all experiments, RBF kernel is used. RBF kernel requires the tuning of two parameters: C and γ. As suggested by Hsu et al. [60], this study used exponential growing sequence of C and γ. Furthermore, all experiments were performed on a desktop computer with the following specification: Windows 7, 64 bits, 8GB RAM, Intel core (TM) i7-4770S CPU @ 3.10GHz. **Tables 1 and 2** shows the parameters used for all experiments. The parameters are similar to the parameters used by authors in [61] and [62].

The following key are used for the Tables: NA-Number of agents, NI-number of instances, GB-global best, APA-Average Prediction Accuracy, FP-False Positive, FN-False Negative, R-Recall, Pr-Precision, FM-F-Measure, T-Time, N_g = Number of generations

Table 1. Parameter used for CISA

Discovery Rate	**Tolerance**	**N_g(Filter)**	**N_g(Wrapper)**	**Beta**
0.25	$1.0e^{-5}$	5	3	1.5

Table 2. Parameter used for BISA

Loudness	**Pulse Rate**	N_g(Filter)	N_g(Wrapper)	**Minimum Frequency**	**Maximum Frequency**
0.5	0.5	5	3	0	2

Table 3. Datasets used for Experiments

Dataset Name	Size	Ham	Spam/Phishing
Dataset 1	4000	3500	Spam: 500 (12.5%)
Dataset 2	4000	3500	Phishing: 500 (12.5%)
Dataset 3	4600	2787	Spam: 1813 (39.4%)
Dataset 4	5000	4508	Credit Card: 492 (9.84%)

3.4 Results and Discussion

Tables 4 - 6 shows the result for KNN_IS and CLUS_IS, for phishing email, credit card fraud and spam email. As shown in both tables, CLUS_IS yielded good classification accuracy, however, its classification speed is poor. KNN_IS yielded better classification speed and accuracy, compared to CLUS_IS. Overall, CLUS_IS produced good classification accuracy, but at the expense of speed. KNN_IS produced better classification accuracy and speed compared to CLUS_IS.

Tables 4 – 6 shows the credit card fraud, spam and phishing email result for the proposed filter-based CSISA and BISA. Also, **Tables 9 – 11** shows the credit card fraud, spam and phishing email result for the proposed wrapper-based CSISA and BISA. As shown in **Tables 4 - 6**, the filter-based techniques correctly classified over 96% credit card transactions, in less than 85 seconds. Moreover, they correctly classified over 99% phishing emails within 50 seconds. Additionally, they correctly classified over 95% spam emails in less than 62 seconds. Also, as shown in the Tables, all the techniques requires a maximum of 10% of the training set to produce the above-mentioned results. Moreover, the filter-based techniques require a maximum of 700 instances to produce robust classification models. As shown in **Tables 9 - 11**, the wrapper-based techniques outperform the filter-based techniques, in terms of classification accuracy. However, the filter-based techniques performed better, in terms of classification speed and storage reduction. Overall, all the techniques produced good results, demonstrating their credibility for instance selection.

Tables 4-6 shows the credit card fraud, spam and phishing email results for CLUS_IS, KNN_IS, standard SVM, BISA and CSISA. As shown in the Tables, the filter-based techniques improved SVM training speed by over 93%, without significantly affecting SVM classification quality. Moreover, they outperform CLUS_IS and KNN_IS, in terms of classification speed and storage reduction. Additionally, **Tables 9-11** shows the credit card fraud, spam and phishing email results produced by standard SVM and the wrapper-based techniques. As shown in the Tables, the wrapper-based techniques improved SVM training speed by over 46%, and simultaneously improved SVM classification accuracy. The wrapper-based techniques also reduced the training dataset by an average of 50%. **Table 7** shows the result for CLUS_IS [27], KNN_IS [6], PSC [57], DROP 3 [26], DROP 5 [26], GCNN [58] , POC-NN [59] and the filter-based BISA and CSISA. As shown in the Table,

the proposed techniques outperform the seven compared techniques, in both classification accuracy and speed.

Tables 8 and 13 reports the classification accuracy, speed and storage reduction percentage produced by standard SVM and the proposed techniques, for 20 UCI datasets. As shown in **Table 8**, for each dataset, the best classification speed is underlined. As reported, the filter-based techniques significantly improved SVM classification speed and storage reduction for all the 20 datasets, without meaningfully affecting SVM classification accuracy. Also, as shown in **Table 13**, for each dataset, the best classification accuracy is underlined. As reported in the table, the wrapper-based techniques consistently produced better predictive accuracy in 75% (15 out of 20) of the datasets, compared to standard SVM. They also produced better classification speed and reduced the training dataset size by an average of 50%. **Table 13** shows the result for ADR-Miner [17] (an existing wrapper-based technique). As shown in the Table, CSISA and BISA outperform ADR-Miner in 90% (9 out of 10) datasets used for evaluation. **Table 12** shows the phishing email result for the proposed techniques and four existing ML-based phishing email detection techniques. As shown in the Table, the wrapper-based techniques outperform three of the four techniques.

Finally, two sample, Z-Test statistical analysis was performed to evaluate the credibility of all the results. The analysis was performed with the primary objective of showing (with 95% confidence level) that the proposed filter and wrapper based techniques are significantly faster than standard SVM. As shown in **Tables 14 and 15**, the filter-based and wrapper-based techniques significantly improve SVM classification speed.

4. Conclusion

SVM is a popular ML algorithm that has been widely applied to classification and regression problems. However, SVM classification speed decreases with increase in dataset size. This paper propose two filter-based and wrapper-based instance selection techniques for improving SVM classification speed and accuracy. The primary difference between the filter and wrapper based techniques is in their objectives. The filter-based techniques are designed with the objective of improving SVM classification speed, and the wrapper-based techniques are designed with the primary objective of improving SVM classification accuracy and speed. The filter-based techniques are very useful for applications (such as video surveillance and intrusion detection) that requires very fast online training of large datasets. Also, the wrapper-based techniques are useful for applications (such as spam email and phishing email classifiers) that are very sensitive to slight drop in classification accuracy.

The proposed techniques are validated on 24 datasets. Initially, they are evaluated on datasets containing three popular e-fraud types: credit card fraud, phishing email and spam email. Furthermore, they are evaluated on datasets containing 21 other problems obtained from UCI dataset repository. Experimental results show that the filter-based techniques excellently improved SVM training speed in 100% of the datasets, without significantly affecting SVM classification quality. Also, results shows that the wrapper-based techniques improved SVM predictive accuracy in 74% of the datasets (17 out of 23), and simultaneously improved SVM training speed. Additionally, experimental result show that

the proposed techniques produced excellent storage reduction and speed-accuracy tradeoff. Finally, two-samples Z-test statistical analysis was performed to evaluate the speed of the proposed techniques, and experimental results reveal that the filter and wrapper based techniques significantly improve SVM classification speed. Overall, as shown in all the results, the proposed filter and wrapper based techniques are very fast, accurate and reliable SVM-based e-fraud detection and instance selection techniques.

Supervised learning algorithms (such as SVM) and other ML algorithms may not produce optimal or accurate results when applied to anonymized datasets. Riyazuddin and Balaram [63] proposed a novel pattern anonymization technique by using feature set partitioning in combination with data restructuring. The proposed technique was predominantly designed to improve the performance of supervised learning algorithms, when applied to anonymized datasets. Data anonymization is an interesting domain, and an avenue for further research.

Table 4. Filter-based Techniques vs existing filter-based techniques for Credit Card Fraud

Technique	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage Reduction
CSISA	96.94	99.20	1.87	14.04	85.96	85.54	84.97	34.72	3.18
BISA	97.40	99.20	1.56	12.14	87.86	87.43	87.07	84.88	7.15
CLUS [27]	98.47	99.4	0.46	11.31	88.69	95.48	91.9	684.06	41.67
KNN [6]	92.5	97.4	7.84	4.41	95.59	58.83	72.76	259.22	11.11
Standard SVM	98.83	99.4	0.29	9.23	90.77	97.07	93.79	2072.99	0

Table 5. Filter-based BISA and CSISA vs existing filter-based techniques for Phishing Email

Technique	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage (%)
CSISA	99.31	100	0.45	2.34	97.66	97.06	97.22	30.54	5.90
BISA	99.43	100	0.33	2.28	97.72	97.79	97.62	45.62	8.92
CLUS [27]	99.53	100	0.23	2.16	97.84	98.47	98.03	337.46	41.67
KNN [6]	99.59	100	0.25	1.56	98.44	98.34	98.3	244.15	5.56
Standard SVM	99.66	100	0.08	2.2	97.8	99.47	98.52	943.24	0

Table 6. Filter-based BISA and CSISA vs existing filter-based techniques for Spam Email

Technique	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage (%)
CSISA	96.31	97.50	3.59	4.40	95.60	79.58	86.60	39.63	5.73
BISA	96.36	97.50	3.57	4.14	95.86	79.52	86.84	46.03	6.27
CLUS [27]	96.44	100	2.61	10.28	89.72	84.41	85.35	311.98	41.67
KNN [6]	95.57	97.5	4.52	3.8	96.2	75.77	84.58	170.38	11.11
Standard SVM	96.66	97.5	3.13	4.8	95.2	81.28	87.62	953.94	0

Table 7. Filter based BISA and CSISA vs Other Techniques for Spambase

Technique	APA(%)	T(s)
BISA	86.71	96.87
CISA	88.15	91.22
KNN_IS [6]	85.59	758.94

CLUS_IS [27]	92.70	7375.75
PSC [57]	71.95	189.57
DROP 3 [26]	78.44	3782.57
DROP 5 [26]	78.72	2226.42
GCNN [58]	73.54	348.56
POC-NN [59]	75.37	735.08

Table 8. Filter-based BISA and CSISA vs Standard SVM for UCI Datasets

Dataset Name	CSISA			BISA			SVM		
	Accuracy	Storage	Time	Accuracy	Storage	Time	Accuracy	Storage	Time
Abalone	52.96	5.58	42.28	53.21	8.68	65.77	55.71	0	2010
Balance Scale	88.71	29.41	29.82	90.52	46.11	49.64	93.71	0	101.1
Breast Tissue	58.10	44.17	7.24	57.70	45.01	7.20	64.6	0	15.98
Bupa	62.21	27.59	14.89	66.56	45.10	25.52	71.56	0	64.81
credit-g	62.21	23.14	14.89	66.56	36.28	25.52	75.95	0	299.9
Cleaveland	59.59	27.06	12.02	60.90	44.39	19.56	63.21	0	53.55
Ecoli	84.06	28.60	16.41	85.82	45.15	24.61	87.36	0	62.1
Glass	61.05	26.76	9.42	63.52	44.48	14.82	65.67	0	33.95
Hungarian	63.34	27.26	14.69	64.03	45.08	24.20	63.86	0	52.12
Iris	94.47	29.65	6.14	94.73	42.84	9.53	95.5	0	21.45
Liver	62.56	27.68	15.96	65.24	45.40	27.58	72.47	0	58.26
Pima Indians	74.25	29.69	42.91	75.03	46.47	70.93	76.92	0	126.7
Post Operative	71.63	55.94	6.90	71.25	55.92	7.04	71.25	0	11.87
Transfusion	77.74	29.84	32.06	78.09	46.62	53.35	78.61	0	135.2
Vertebral-3c	82.29	27.55	14.82	84.16	44.92	21.5	85.61	0	53.51
Voting	94.09	28.58	18.73	94.93	45.47	30.86	95.77	0	83.07
Waveform	82.81	4.66	51.03	83.89	7.28	79.05	86.98	0	2501
Wine	96.29	26.64	5.29	97.59	43.93	8.75	97.47	0	32.58
Yeast	55.48	18.19	50.06	57.39	28.22	80.07	59.45	0	306
Zoo	90.40	44.66	7.32	91.40	45.54	7.66	95	0	17.74
Average	73.71	28.13	20.64	75.13	40.64	32.66	77.83	0	302.04

Table 9. Wrapper-based BISA and CSISA vs standard SVM for Credit Card Fraud

Technique	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage (%)
CSISA	98.83	99.60	0.26	9.57	90.43	97.54	93.75	649.95	44.1
BISA	98.84	99.60	0.29	9.07	90.93	97.26	93.92	828.73	50.01
Standard SVM	98.83	99.4	0.29	9.23	90.77	97.07	93.79	2072.99	0

Table 10. Wrapper-based BISA and CSISA vs standard SVM for Phishing Email

Technique	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage (%)
CSISA	99.62	100	0.13	2.18	97.82	99.14	98.35	378.12	47.83
BISA	99.62	100	0.13	2.14	97.86	99.1	98.37	409.69	50.1
Standard SVM	99.66	100	0.08	2.2	97.8	99.47	98.52	943.24	0

Table 11. Wrapper-based BISA and CSISA vs standard SVM for Spam Email

Technique	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage (%)
CSISA	96.92	99.25	2.89	4.4	95.6	82.73	88.56	454.91	46.21
BISA	96.80	97.75	3.11	3.86	96.14	81.56	88.21	442.32	50.03
Standard SVM	96.66	97.5	3.15	4.66	95.34	81.25	87.67	853.15	0

Table 12. Wrapper-based BISA and CSISA vs Existing Techniques

Technique	APA(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)
CSISA	99.62	0.13	2.18	97.82	99.14	98.35
BISA	99.62	0.13	2.14	97.86	99.1	98.37
Akinyelu and Adewumi [4]	99.70	0.06	2.50	97.50	99.47	98.45
Andre et al [3]	99.13	0.20	6.39	93.61	98.26	95.88
Fette et al. [51]	99.49	0.13	3.62	96.38	98.92	97.64
Zhang and Yuan [52]	95.51	-	-	96.18	95.25	95.71

Table 13. Wrapper-based BISA and CSISA vs Standard SVM and ADR-Miner for UCI Datasets

Dataset Name	CSISA			BISA			ADR-Miner			Standard SVM		
	Accu	Stor	Time	Accu	Stor	Time	Accu	Stor	Time	Accu	Stor	Time
Abalone	56.7	37.9	746.0	56.4	50.0	1014.0	-	-	-	55.7	0	2010
Balance Scale	91.5	42.3	74.0	91.5	50.5	81.9	-	-	-	93.7	0	101.1
Breast Tissue	69.5	47.4	14.2	67.6	50.7	13.8	60.6	24.0	-	64.6	0	15.98
Bupa	69.9	40.0	39.3	69.8	49.9	44.5	-	-	-	71.6	0	64.81
credit-g	75.9	39.7	150.2	75.8	49.8	179.6	74.1	19.3		76.0	0	299.9
Cleaveland	64.9	38.6	33.5	64.1	49.8	40.0	-	-	-	63.2	0	53.55
Ecoli	89.3	44.5	41.6	88.1	50.3	46.3	81.3	21.3	-	87.5	0	62.1
Glass	71.1	39.6	24.9	69.7	50.4	29.3	69.6	31.4	-	65.7	0	33.95
Hungarian	67.8	37.2	33.5	66.3	49.8	41.8	-	-	-	63.9	0	52.12
Iris	97.7	46.7	17.0	96.6	49.7	17.8	92.6	42.1	-	95.5	0	21.45
Liver	71.5	39.3	41.7	70.2	50.6	44.6	58.6	17.6	-	72.5	0	58.26
Pima Indians	78.4	38.7	102.4	77.2	49.7	111.6	-	-	-	76.9	0	126.7
Post Operative	71.5	56.4	12.1	72.1	56.7	11.6	-	-	-	71.3	0	11.87
Transfusion	79.2	38.2	84.8	79.5	50.3	95.9	72.3	21.9	-	78.6	0	135.2
vertebral-3c	87.7	42.3	35.0	86.8	49.9	37.8	83.6	23.3	-	85.6	0	53.31
Voting	96.5	43.9	49.2	96.5	50.2	58.0	95.5	12.0	-	95.8	0	83.07
Waveform	86.8	39.8	1300	86.8	50.0	1597	-	-	-	87.0	0	2501
Wine	97.8	49.1	17.7	97.9	50.2	17.4	-	-	-	97.5	0	32.58
Yeast	60.9	40.7	185.5	60.9	50.0	209.5	-	-	-	59.5	0	306
Zoo	97.0	50.3	12.3	96.6	50.4	13.0	98.8	52.8	-	95.0	0	17.74
Average	79.6	42.5	144.3	78.8	50.0	177.1	-	-	-	78.5	0	288.99

Key: Accu: Accuracy, Stor: Storage

Table 14. Statistical Analysis: Filter-based BISA and CSISA

Technique	e-Fraud Type	Number of Samples	α = 0.05 Critical Value = 1.959963985
CSISA vs CLUS [27]	Credit Card Fraud	100	155.2301779
CSISA vs KNN [6]	Credit Card Fraud	100	134.1201855
CSISA vs Standard SVM	Credit Card Fraud	100	126.473953
BISA vs CLUS [27]	Credit Card Fraud	100	140.1926725
BISA vs KNN [6]	Credit Card Fraud	100	92.2637861
BISA vs Standard SVM	Credit Card Fraud	100	123.1792182
CSISA vs CLUS [27]	Phishing Email	100	33.2293053
CSISA vs KNN [6]	Phishing Email	100	94.24803824
CSISA vs Standard SVM	Phishing Email	100	284.6893032
BISA vs CLUS [27]	Phishing Email	100	31.24861442
BISA vs KNN [6]	Phishing Email	100	74.76322296
BISA vs Standard SVM	Phishing Email	100	257.0614655
CSISA vs CLUS [27]	Spam Email	100	104.5245697
CSISA vs KNN [6]	Spam Email	100	48.92006215
CSISA vs Standard SVM	Spam Email	100	298.3137267
BISA vs CLUS [27]	Spam Email	100	101.3975132
BISA vs KNN [6]	Spam Email	100	160.4279588
BISA vs Standard SVM	Spam Email	100	294.8151401

Table 15. Statistical Analysis: Wrapper-based BISA and CSISA

Technique	e-Fraud Type	Number of Samples	α = 0.05 Critical Value = 1.959963985
CSISA vs Standard SVM	Credit Card Fraud	100	72.31003626
BISA vs Standard SVM	Credit Card Fraud	100	70.89500213
CSISA vs Standard SVM	Phishing Email	100	85.1808948
BISA vs Standard SVM	Phishing Email	100	90.91963131
CSISA vs Standard SVM	Spam Email	100	75.31108681
BISA vs Standard SVM	Spam Email	100	87.91818266

Reference

[1] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine learning*, vol. 20, no. 3, pp. 273-297, September, 1995. Article (CrossRef Link).

[2] B. Yashvantrai Vyas, R. P. Maheshwari, and B. Das, "Pattern Recognition Application of Support Vector Machine for Fault Classification of Thyristor Controlled Series Compensated Transmission Lines," *Journal of The Institution of Engineers (India): Series B*, vol. 97, no. 2, pp. 175-183, June, 2016. Article (CrossRef Link).

[3] A. Bergholz, J. H. Chang, G. Paaß, F. Reichartz, and S. Strobel, "Improved Phishing Detection using Model-Based Features," in *Proc. of the Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, pp. 1-27, August 21-22, 2008. Article (CrossRef Link).

[4] A. A. Akinyelu and A. O. Adewumi, "Classification of phishing email using random forest machine learning technique," *Journal of Applied Mathematics*, vol. 2014, Article ID 425731, 6 pages, April, 2014. Article (CrossRef Link).

[5] E. Kremic and A. Subasi, "Performance of random forest and SVM in face recognition," *Int. Arab J. Inf. Technol.*, vol. 13, no. 2, pp. 287-293, March, 2016. Article (CrossRef Link).

[6] N. Panda, E. Y. Chang, and G. Wu, "Concept boundary detection for speeding up SVMs," in *Proc. of the 23rd international conference on Machine learning*, pp. 681-688, June 25 - 29, 2006. Article (CrossRef Link).

[7] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and J. Kittler, "A review of instance selection methods," *Artificial Intelligence Review*, vol. 34, no. 2, pp. 133-143, August, 2010. Article (CrossRef Link).

[8] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations," *The Journal of Machine Learning Research*, vol. 2, pp. 243-264, December, 2002. Article (CrossRef Link).

[9] B. L. Narayan, C. A. Murthy, and S. K. Pal, "Maxdiff kd-trees for data condensation," *Pattern Recognition Letters*, vol. 27, no. 3, pp. 187-200, February, 2006. Article (CrossRef Link).

[10] H. Liu and H. Motoda, "On Issues of Instance Selection," *Data Mining and Knowledge Discovery*, vol. 6, no. 2, pp. 115-130, April, 2002. Article (CrossRef Link).

[11] J. C. Bezdek and L. I. Kuncheva, "Nearest prototype classifier designs: An experimental study," *International Journal of Intelligent Systems*, vol. 16, no. 12, pp. 1445-1473, December, 2001. Article (CrossRef Link).

[12] V. Cerveron and F. J. Ferri, "Another move toward the minimum consistent subset: a tabu search approach to the condensed nearest neighbor rule," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 31, no. 3, pp. 408-413, June, 2001. Article (CrossRef Link).

[13] J. A. Olvera-López, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "Sequential search for decremental edition," in *Proc. of International Conference on Intelligent Data Engineering and Automated Learning*, pp. 280-285, July 6-8, 2005. Article (CrossRef Link).

[14] L. I. Kuncheva, "Fitness functions in editing k-NN reference set by genetic algorithms," *Pattern Recognition*, vol. 30, no. 6, pp. 1041-1049, June, 1997. Article (CrossRef Link).

[15] J. R. Cano, F. Herrera, and M. Lozano, "Stratification for scaling up evolutionary prototype selection," *Pattern Recognition Letters*, vol. 26, no. 7, pp. 953-963, May, 2005. Article (CrossRef Link).

[16] S. García, J. R. Cano, and F. Herrera, "A memetic algorithm for evolutionary prototype selection: A scaling up approach," *Pattern Recognition*, vol. 41, no. 8, pp. 2693-2709, August, 2008. Article (CrossRef Link).

[17] I. M. Anwar, K. M. Salama, and A. M. Abdelbar, "Instance selection with ant colony optimization," *Procedia Computer Science*, vol. 53, pp. 248-256, January, 2015. Article (CrossRef Link).

[18] U. Garain, "Prototype reduction using an artificial immune model," *Pattern Analysis and Applications*, vol. 11, no. 3, pp. 353-363, September, 2008. Article (CrossRef Link).

[19] M. Behdad, L. Barone, M. Bennamoun, and T. French, "Nature-inspired techniques in the context of fraud detection," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1273-1290, November, 2012. Article (CrossRef Link).

[20] KrebsOnSecurity. (2015), "FBI: $1.2B Lost to Business Email Scams". available at: http://krebsonsecurity.com/2015/08/fbi-1-2b-lost-to-business-email-scams/ (accessed 14-September - 2016).
[21] T. N. Report. (2016, 01-August-2017). Card Fraud Worldwide. 12. Available: https://www.nilsonreport.com/upload/content_promo/The_Nilson_Report_10-17-2016.pdf
[22] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," *Data mining and knowledge discovery*, vol. 6, no. 2, pp. 153-172, April, 2002. Article (CrossRef Link).
[23] T. Reinartz, "A Unifying View on Instance Selection," *Data Mining and Knowledge Discovery*, vol. 6, no. 2, pp. 191-210, April, 2002. Article (CrossRef Link).
[24] J. Yang and S. Olafsson, "Optimization-based feature selection with adaptive instance sampling," *Computers & Operations Research*, vol. 33, no. 11, pp. 3088-3106, November, 2006. Article (CrossRef Link).
[25] C.-F. Tsai, W. Eberle, and C.-Y. Chu, "Genetic algorithms in feature and instance selection," *Knowledge-Based Systems*, vol. 39, pp. 240-247, February, 2013. Article (CrossRef Link).
[26] D. R. Wilson and T. R. Martinez, "Reduction Techniques for Instance-Based Learning Algorithms," *Machine Learning*, vol. 38, no. 3, pp. 257-286, March, 2000. Article (CrossRef Link).
[27] J. Chen, C. Zhang, X. Xue, and C.-L. Liu, "Fast instance selection for speeding up support vector machines," *Knowledge-Based Systems*, vol. 45, pp. 1-7, June, 2013. Article (CrossRef Link).
[28] H. Lei and V. Govindaraju, "Speeding up multi-class SVM evaluation by PCA and feature election," in *Proc. of the Workshop on Feature Selection for Data Mining:Interfacing Machine Learning and Statistics* Newport Beach, CA, April 22, 2005. Article (CrossRef Link).
[29] A. O. Adewumi and M. M. Ali, "A multi-level genetic algorithm for a multi-stage space allocation problem," *Mathematical and Computer Modelling*, vol. 51, no. 1, pp. 109-126, January, 2010. Article (CrossRef Link).
[30] T. R. Jensen and B. Toft, "Graph coloring problems," vol. 39, 2011. Article (CrossRef Link).
[31] S. Chetty and A. O. Adewumi, "Three new stochastic local search metaheuristics for the annual crop planning problem based on a new irrigation scheme," *Journal of Applied Mathematics*, vol. 2013, Article ID 158538, 14 pages, 2013., May, 2013. Article (CrossRef Link).
[32] O. A. Adewumi and A. A. Akinyelu, "A hybrid firefly and support vector machine classifier for phishing email detection," *Kybernetes*, vol. 45, no. 6, pp. 977-994, June, 2016. Article (CrossRef Link).
[33] X.-S. Yang and X. He, "Firefly algorithm: recent advances and applications," *International Journal of Swarm Intelligence*, vol. 1, no. 1, pp. 36-50, January, 2013. Article (CrossRef Link).
[34] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. of IEEE international conference on neural networks*, vol. 4, no. 2, pp. 1942-1948, November, 1995. Article (CrossRef Link).
[35] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671-680, May, 1983. Article (CrossRef Link).
[36] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proc. of World Congress on Nature & Biologically Inspired Computing, 2009. NaBIC 2009.* , pp. 210-214, December 9-11, 2009. Article (CrossRef Link).
[37] X.-S. Yang, "A New Metaheuristic Bat-Inspired Algorithm," in *Proc. of Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, J. R. González, D. A. Pelta, C. Cruz, G. Terrazas, and N. Krasnogor, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 65-74, 2010. Article (CrossRef Link).

[38] D. Rodrigues, L. A. M. Pereira, R. Y. M. Nakamura, K. A. P. Costa, X.-S. Yang, A. N. Souza, *et al.*, "A wrapper approach for feature selection based on Bat Algorithm and Optimum-Path Forest," *Expert Systems with Applications*, vol. 41, no. 5, pp. 2250-2258, April, 2014. Article (CrossRef Link).

[39] S. A. Medjahed, T. A. Saadi, A. Benyettou, and M. Ouali, "Binary cuckoo search algorithm for band selection in hyperspectral image classification," *IAENG International Journal of Computer Science*, vol. 42, no. 3, pp. 183-191, July, 2015. Article (CrossRef Link).

[40] A. M. Taha, A. Mustapha, and S.-D. Chen, "Naive bayes-guided bat algorithm for feature selection," *The Scientific World Journal*, vol. 2013, Article ID 325973, 9 pages, 2013., December, 2013. Article (CrossRef Link).

[41] E. Emary, W. Yamany, and A. E. Hassanien, "New approach for feature selection based on rough set and bat algorithm," in *Proc. of 9th International Conference on Computer Engineering & Systems (ICCES)*, pp. 346-353, December 22-23, 2014. Article (CrossRef Link).

[42] M. A. Laamari and N. Kamel, "A hybrid bat based feature selection approach for intrusion detection," in *Proc. of Bio-Inspired Computing-Theories and Applications*, ed: Springer, pp. 230-238, 2014. Article (CrossRef Link).

[43] R. R Rajalaxmi, "A Hybrid Binary Cuckoo Search and Genetic Algorithm for Feature Selection in Type-2 Diabetes," *Current Bioinformatics*, vol. 11, no. 4, pp. 490-499, September, 2016. Article (CrossRef Link).

[44] S. Mousavirad and H. Ebrahimpour-Komleh, "Wrapper feature selection using discrete cuckoo optimization algorithm," *International Journal of Mechatronics Electrical, and Computer Engineering*, vol. 4, no. 11, pp. 709-721, April, 2014. Article (CrossRef Link).

[45] K. Bache and M. Lichman. (2013), "UCI machine learning repository". available at: http://archive.ics.uci.edu/ml (accessed 12-May-2017).

[46] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, April, 2011. Article (CrossRef Link).

[47] P. Graham., "A Plan for Spam," 2002. available at: http://www.paulgraham.com/spam.html (accessed 04-August-2016).

[48] R. Shams and R. E. Mercer, "Classifying Spam Emails Using Text and Readability Features," in *Proc. of IEEE 13th International Conference on Data Mining*, pp. 657-666, December 7-10, 2013. Article (CrossRef Link).

[49] R. Duncan. "A Simple Guide to HTML," available at: http://www.simplehtmlguide.com/whatisht- ml.php (accessed 13-September-2016).

[50] A. Almomani, T.-C. Wan, A. Altaher, A. Manasrah, E. ALmomani, M. Anbar, *et al.*, "Evolving fuzzy neural network for phishing emails detection," *Journal of Computer Science*, vol. 8, no. 7, p. 1099, July, 2012. Article (CrossRef Link).

[51] I. Fette, N. Sadeh, and A. Tomasic, "Learning to detect phishing emails," in *Proc. of the 16th international conference on World Wide Web*, Banff, AB, Canada, pp. 649-656, May 8-12, 2007. Article (CrossRef Link).

[52] N. Zhang and Y. Yuan, "Phishing Detection Using Neural Network," CS229 lecture notes. Article (CrossRef Link).

[53] C. Group., "SpamAssassin Data," 2006. available at: http://www.csmining.org/index.php/spam-assassin-datasets.html (accessed 05-August-2014).

[54] J. Nazario., "Phishing Corpus," 2006. available at: http://monkey.org/jose/wiki/doku.php?id=PhishingCorpus (accessed 27-April-2015).

[55] A. Asuncion and D. Newman., "UCI Machine Learning Repository," 2007. available at: http://archive.ics.uci.edu/ml/datasets.html (accessed 15-August-2016).

[56] Andrea., "Credit Card Fraud Detection," 2016. available at: https://www.kaggle.com/dal-pozz/creditcardfraud (accessed 12-December-2016).

[57] J. A. Olvera-López, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "A new fast prototype selection method based on clustering," *Pattern Analysis and Applications*, vol. 13, no. 2, pp. 131-141, May, 2010. Article (CrossRef Link).
[58] C. Chien-Hsing, K. Bo-Han, and C. Fu, "The Generalized Condensed Nearest Neighbor Rule as A Data Reduction Method," in *Proc. of 18th International Conference on Pattern Recognition (ICPR'06)*, pp. 556-559, August 20-24, 2006. Article (CrossRef Link).
[59] T. Raicharoen and C. Lursinsap, "A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (POC-NN) algorithm," *Pattern Recognition Letters*, vol. 26, no. 10, pp. 1554-1567, July, 2005. Article (CrossRef Link).
[60] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification. Tech. rep., Department of Computer Science, National Taiwan University.," no. 1-16, 2003. Article (CrossRef Link).
[61] X.-S. Yang. (2010), "Cuckoo Search (CS) Algorithm," available at: https://www.mathworks.c-om/matlabcentral/fileexchange/29809-cuckoo-search-cs-algorithm/content/cuckoo_search.m (accessed 11-September-2016).
[62] X.-S. Yang. (2015), "Bat Algorithm". available at: https://www.mathworks.com/matlabcentral/ fileexchange/37582-bat-algorithm--demo-/content/bat_algorithm.m (accessed 11-September-2016).
[63] M. Riyazuddin and V. V. S. S. S. Balaram, "Pattern Anonymization: Hybridizing Data Restructure with Feature Set Partitioning for Privacy Preserving in Supervised Learning," in *Proc. of the First International Conference on Computational Intelligence and Informatics : ICCII 2016*, S. C. Satapathy, V. K. Prasad, B. P. Rani, S. K. Udgata, and K. S. Raju, Eds., ed Singapore: Springer Singapore, pp. 603-614, 2017. Article (CrossRef Link).

Akinyelu A. Ayobami received his B.Sc. (Hons.) degree in Computer Science, from the Federal University of Technology, Akure, Nigeria. He also received his MSc. Degree from the University of KwaZulu-Natal, Durban, South Africa, where he is currently pursuing his PhD degree in Computer Science. His current research interest include: artificial intelligence and machine learning, with a focus on solutions to e-fraud detection, big data problems and optimization problems.

Aderemi O. Adewumi received the B.Sc. and M.Sc. degrees in computer science from the University of Lagos, Nigeria, and the Ph.D. degree in computational and applied mathematics from the University of Witwatersrand, South Africa, with specialty in optimization, computational intelligence and machine learning. He is currently with the University of KwaZulu-Natal, Durban, South Africa, where he leads the Optimization and Modeling Research Group in the School of Mathematics, Statistics and Computer Science. His current research interests include optimization and artificial intelligence, with a particular interest in computational intelligence, intelligent learning and (meta) heuristics solutions to real-world global optimization problems.