Intelligent Instance Selection Techniques for Support Vector Machine Speed Optimization with Application to e-Fraud Detection

by

AKINYELU Ayobami Andronicus

Student No. 213574026

Submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy in Computer Science

at the

School of Mathematics, Statistics and Computer Science
University of KwaZulu-Natal
Durban, South Africa

September, 2017

UNIVERSITY OF KWAZULU-NATAL

COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE

DECLARATION

The research described in this thesis was performed at the University of KwaZulu-Natal under the supervision of Prof.. A. O. Adewumi. I hereby declare that all materials incorporated in this thesis are my own original work except where acknowledgement is made by name or in the form of a reference. The work contained herein has not been submitted in part or whole for a degree at any other university.

Signed:

[illegible]

____

Akinyelu Ayobami Andronicus

Date: September 2017

As the candidate’s supervisor, I have approved the Thesis for submission

Signed:

____

Prof. A. O. Adewumi

Date: September 2017

UNIVERSITY OF KWAZULU-NATAL

COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE

DECLARATION - PLAGIARISM

I, ___Akinyelu Ayobami Andronicus, declare that:

1. The research reported in this thesis, except where otherwise indicated, is my original research.
2. This thesis has not been submitted for any degree or examination at any other University.
3. This thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
4. This thesis does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
    a. Their words have been re-written but the general information attributed to them has been referenced
    b. Where their exact words have been used, then their writing has been placed in italics and inside quotation marks, and referenced.
5. This thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the References sections.

Signed:

[illegible]

Akinyelu Ayobami Andronicus

Dedication

My PhD is dedicated to the Almighty God, the Author and Finisher of my Faith. He is solely responsible for the success of this research work.

Acknowledgements

My first acknowledgement goes to the Almighty God, the author and finisher of this research; all glory and honor be to His holy name. Secondly, I am very grateful to my beloved father and mother: Pastor and Mrs Akinyelu, for their priceless and cherished support throughout this research. Words cannot express how thankful I am; they are the best parents in the whole wide world! Thank you very much! Furthermore, I am deeply thankful to my darling siblings: Opeyemi, Jude and Ebenezer. They were there for me throughout my PhD research; thank you very much!

My profound and special gratitude goes to my amiable supervisor, Professor Aderemi Adewumi. Thank you very much for your treasured contributions, advice, comments and guidance. I am truly very grateful! Furthermore, I want to extend my deepest appreciation to Dr Martins Arasomwan and Dr Absalom Ezugwu; my research greatly benefited from their excellent contributions and assistance, thank you very much. Additionally, I am very grateful to my beloved pastors, Pastor Micheal Olusanya and Pastor Joseph Adesina and mother, Mrs Adewumi, for their unceasing care and support; God will surely reward three of you in multiple folds. Moreover, I am very thankful to my dear friend, Jerimiah Ogunniyi and my other brothers and sisters in Deeper Life Campus Fellowship, Westville Campus and other Campuses. My time with you all made remarkable impact in my life. May God reward you all!

My acknowledgement will be incomplete if I skip all the staffs of the school of Mathematics, Statistics and Computer Science. Special thanks goes to Mrs Charmaine Magwaza, Mrs Selvie Moodley, Mrs Shereen, Mr Greenwood and Mr Evans. Thank you very much for all your support and assistance, I am sincerely grateful. Finally, I am very thankful to the University of KwaZulu-Natal for providing a conducive environment and giving me the golden opportunity to pursue my PhD studies in this amazing university. Thank you very much.

Table of Contents

Abstract

Decision-making is a very important aspect of many businesses. There are grievous penalties involved in wrong decisions, including financial loss, damage of company reputation and reduction in company productivity. Hence, it is of dire importance that managers make the right decisions. Machine Learning (ML) simplifies the process of decision making: it helps to discover useful patterns from historical data, which can be used for meaningful decision-making. The ability to make strategic and meaningful decisions is dependent on the reliability of data. Currently, many organizations are overwhelmed with vast amounts of data, and unfortunately, ML algorithms cannot effectively handle large datasets. This thesis therefore proposes seven filter-based and five wrapper-based intelligent instance selection techniques for optimizing the speed and predictive accuracy of ML algorithms, with a particular focus on Support Vector Machine (SVM). Also, this thesis proposes a novel fitness function for instance selection. The primary difference between the filter-based and wrapper-based technique is in their method of selection. The filter-based techniques utilizes the proposed fitness function for selection, while the wrapper-based technique utilizes SVM algorithm for selection.

The proposed techniques are obtained by fusing SVM algorithm with the following Nature Inspired algorithms: flower pollination algorithm, social spider algorithm, firefly algorithm, cuckoo search algorithm and bat algorithm. Also, two of the filter-based techniques are boundary detection algorithms, inspired by edge detection in image processing and edge selection in ant colony optimization. Two different sets of experiments were performed in order to evaluate the performance of the proposed techniques (wrapper-based and filter-based). All experiments were performed on four datasets containing three popular e-fraud types: credit card fraud, email spam and phishing email. In addition, experiments were performed on 20 datasets provided by the well-known UCI data repository. The results show that the proposed filter-based techniques excellently improved SVM training speed in 100% (24 out of 24) of the datasets used for evaluation, without significantly affecting SVM classification quality. Moreover, experimental results also show that the wrapper-based techniques consistently improved SVM predictive accuracy in 78% (18 out of 23) of the datasets used for evaluation and simultaneously improved SVM training speed in all cases. Furthermore, two different statistical tests were conducted to further validate the credibility of the results: Freidman’s test and Holm’s post-hoc test. The statistical test results reveal that the proposed filter-based and wrapper-based techniques are significantly faster, compared to standard SVM and some existing instance selection techniques, in all cases. Moreover, statistical test results also reveal that Cuckoo Search Instance Selection Algorithm outperform all the proposed techniques, in terms of speed.

Overall, the proposed techniques have proven to be fast and accurate ML-based e-fraud detection techniques, with improved training speed, predictive accuracy and storage reduction. In real life application, such as video surveillance and intrusion detection systems, that require a classifier to be trained very quickly for speedy classification of new target concepts, the filter-based techniques provide the best solutions; while the wrapper-based techniques are better suited for applications, such as email filters, that are very sensitive to slight changes in predictive accuracy.

List of Tables

List of Abbreviations

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
ACOISA	Ant Colony Optimization Instance Selection Algorithm
AI	Artificial Intelligence
AIS	Artificial Immune System
ANN	Artificial Neural Network
APWG	Anti-Phishing Working Group
BA	Bat Algorithm
BISA	Bat Instance Selection Algorithm
CSA	Cuckoo Search Algorithm
CSISA	Cuckoo Search Instance Selection Algorithm
EA	Evolutionary Algorithm
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
SVM	Support Vector Machine
ML	Machine Learning
NI	Nature Inspired
FFA	Firefly Algorithm
FFISA	Firefly Instance Selection Algorithm
FPA	Flower Pollination Algorithm
FPISA	Flower Pollination Instance Selection Algorithm
GPU	Graphical Processing Unit

HMM	Hidden Marcov Model
IBL	Instance Based Learning
IG	Information Gain
IP	Internet Protocol
KKT	Karush-Kuhn-Tucker
LDA	Latent Dirichlet Allocation
MLP	Multilayer Perceptron
NB	Naïve Bayes
NN	Nearest Neighbor
OTP	One Time Password
PCA	Principal Component Analysis
PIN	Personal Information Number
PSO	Particle Swarm Optimization
QP	Quadratic Programming
RBF	Radial Basis Function
RFE	Recursive Feature Elimination
RST	Rough set theory
SSA	Social Spider Algorithm
SSISA	Social Spider Instance Selection Algorithm
SMO	Sequential Minimal Optimization
TF	Term Frequency
UCI	University of California Irvine
URL	Uniform Resource Locator

List of Algorithm

List of Algorithm Notations

ACA ← *average classification accuracy*

ANN ← *Artificial Neural Network*

best ← *largest distance*

CA ← *classification accuracy*

CB ←*Current Best*

Dim← *Dimension size for each flower*

dist[,]← *Distance*

E← *edge*

FT← *User-defined Fitness Threshold*

G(x)← *Fitness Function*

GB← *Global Best*

GBV← *Global Best Vibration for from entire population*

HV ← *heuristic value*

I← *edge Instances*

IM ←*Instance mask*

K ← *number of k nearest neighbours*

L ← *Light intensity of firefly*

LF← *Levy Flight.*

MaxG ← *maximum generation*

Min ←*Minimum number of selected instances*

N ← *size of the entire training dataset*

newCA← *new Classification Accuracy*

NF ← *number of folds for SVM cross validation*

NFF← Number of fireflies

NI ←Number of Iterations

NL ← neighborhood list

NR ← neighborhood range

NRuns ← number of runs for SVM cross validation

NS ←Number of Selected Instances

NSub ← size of training subset

P ← population size

Pm← User defined probability for changing spider mask

PR ←Pulse Rate

PS← Probability Switch

Tot← Total number of times, each spider changes its target vibration

TS ←Training Subset

TV← Target Vibration

VI← Vibration Intensity

Vote ← vote for each instance.V is is array of size N

X ← Dataset

List of Figures

List of Publications

Articles in peer review journal (ISI)

1. Akinyelu, Andronicus A., and Aderemi O. Adewumi. "Improved Instance Selection Methods for Support Vector Machine Speed Optimization." Security and Communication Networks 2017 (2017).

2. 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, 2016.

Articles accepted for peer review journal (ISI)

1. Adewumi, Aderemi O., and Andronicus A. Akinyelu. "A survey of machine-learning and nature-inspired based credit card fraud detection techniques." International Journal of System Assurance Engineering and Management (2016): 1-17.

2. A. A. Akinyelu and A. O. Adewumi (2016), “On the Performance of Cuckoo Search and Bat Algorithms Based Instance Selection Techniques for SVM Speed Optimization with Application to e-Fraud Detection”, Submitted to KSII Transactions on Internet and Information Systems

Articles ready for peer review journal

1. A. A. Akinyelu and A. O. Adewumi, “Ant Colony Optimization Edge Selection for Support Vector Machine Speed Optimization”

2. A. A. Akinyelu and A. O. Adewumi, “Flower Pollination Algorithm and Social Spider Algorithm for SVM Speed Optimization with Application to e-Fraud Detection”

3. A. A. Akinyelu and A. O. Adewumi, “A Survey of ML-Based and NI-Based Spam Email Detection Techniques”

4. A. A. Akinyelu and A. O. Adewumi, “Phishing Detection based on Machine Learning and Nature Inspired techniques: A Survey”

Chapter 1
Introduction

Since the invention of integrated circuits and computer chips, the world has experienced a global spread of information. The world is surrounded by large volumes of data produced by different sources, including pictures, videos, emails, websites and Internet. Moreover, data contain very useful information for decision-making; hence, robust information extraction techniques are highly required. Machine Learning (ML) algorithms are very useful techniques for information extraction. They can effectively extract relevant patterns from data, analyze the patterns and turn them into meaningful information for decision-making. Many ML algorithms exist; however, this research focused on Support Vector Machine (SVM).

SVM is a supervised, statistics-based ML algorithm, that is widely used to solve many real-world pattern recognition and data mining problems [1], such as parking space problems [2], hostel allocation problems [3] and email classification [4]. However, SVM training complexity increases as problem size and number of classes increases [1]. Its training time is $O(n^2)$, where n refers to number of training instances [5]. This is a major concern, because several real-world applications require the fast processing of large datasets. Many SVM speed optimization techniques have been proposed in the body of literature, and most of these techniques tackle optimization from different approaches, including instance selection, parameter optimization, and feature selection. Instance selection techniques aim to optimize speed by removing irrelevant and superfluous instances from a dataset. Feature selection techniques aim to optimize speed by removing extraneous features from a dataset. Parameter optimization techniques aim to optimize speed by selecting optimal parameters from a list of parameter values. Among these three approaches, instance selection is one of the most efficient [6].

Instance selection techniques are used to minimize SVM training time by discarding superfluous and harmful instances from a training set. Superfluous instances are instances that contribute negligibly to the classification accuracy of a classifier, while harmful instances are instances that lead to increased FP and FN rates [7]. Superfluous and harmful instances contribute less to SVM prediction process [7], hence discarding them does not have a negative impact on the SVM training result [8]. Some instance selection techniques have been proposed in the literature, and the majority of these techniques are based on the k Nearest Neighbour (NN) classifier [6]. Some of the techniques are based on k-d trees [9], clustering [10, 11], tabu search [12] and sequential search [13]. However, very few techniques have 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 research proposes intelligent instance selection techniques for improving SVM training speed and predictive accuracy.

Some sensitive applications requires a classifier to be trained very quickly in order to enable the classifier to identify new target concepts [8]. Moreover, this application requires the classifier to be trained on large training sets. Examples of such applications include video surveillance and intrusion detection. For this kind of applications, SVM training time can be unacceptably high, which renders SVM ineffectual. Furthermore, even in applications where training can be performed offline (such as email detection systems), if the size of the training data or number of classes is large, then SVM computational complexity will be intolerable [8]. Hence, this thesis proposes seven filter-based instance selection techniques for improving the training speed of SVMs. Five of the techniques are based on recent NI algorithms, including Flower Pollination Algorithm (FPA), Cuckoo Search Algorithm (CSA), Firefly Algorithm (FFA), Social Spider Algorithm (SSA) and Bat Algorithm (BA). The remaining two techniques are inspired by edge detection in image processing and edge selection in ACO, respectively. The proposed filter-based techniques are very useful when processing massive datasets with limited storage space. In addition, some applications (such as spam or phishing email classifiers) are very sensitive to a slight drop in classification accuracy. In these applications, classification accuracy is of greater importance, compared to classification speed. For example, misclassification of one important email can lead to a colossal loss of money or loss of business opportunities. Therefore, this thesis also proposes five wrapper-based instance selection techniques for improving SVM predictive accuracy and training speed.

The proposed techniques have been validated on 24 different datasets. Initially, they were validated on datasets containing three popular e-fraud types: credit card fraud, phishing email and spam email. Also, they were also validated on 20 datasets that were provided by University of

California's Irvine (UCI) dataset repository [19]. Experimental results produced by the techniques show improvement in SVM training speed and predictive accuracy. Moreover, the results show that NI algorithms are fast and efficient instance selection techniques. Additionally, the results revealed that the proposed techniques are excellent SVM-based e-fraud detection techniques.

1.1 Background Information

This section provides general information on some basic concepts that are specific to this study. Particularly, this section provides background information on ML and SVM. Moreover, this section provides background information on instance selection techniques.

1.1.1 Machine learning

Arthur Samuel, in 1959, defined ML as a "field of study that gives computers the ability to learn without being explicitly programmed" [20]. ML evolved primarily from computer science and artificial intelligence (AI), and also from other fields, including: applied mathematics, pattern recognition and computational learning theory [21]. ML algorithms are generally used to solve problems involving automatic classification of data (such as e-fraud detection) [22]. They are capable of analyzing contents in a dataset and extracting unknown or concealed patterns from the dataset. ML-based systems are more reliable than many traditional signature-based systems [23]. Signature-based systems are static in nature and also vulnerable to new (or zero day) threats. This is because signature-based systems rely on signatures stored in updatable databases. On the contrary, ML-based cybersecurity systems have the ability to discover new cyber-attacks in real-time, and thus produce better prediction accuracy compared to signature-based systems [23]. Additionally, ML-based systems are easier to maintain than signature-based systems that use complex data structures. This is because, with ML algorithms, compact, simpler and easily maintainable models can be constructed [23]. ML is divided into different classes, including supervised learning (such as SVM, Naïve Bayes (NB), Random Forest (RF); unsupervised learning (such as K-means, Hidden Markov Model (HMM); semi-supervised learning and reinforcement learning [24]. This thesis proposes intelligent instance selection techniques for improving the training speed and predictive accuracy of ML algorithms, with a particular focus on SVM. SVM is one of the well-known ML algorithms used to handle classification and regression problems.

1.1.2 Support vector machine

SVM is one of the prevalent supervised ML algorithms, with a robust theoretical background, excellent classification accuracy and good generalization performance [25, 26]. SVM use hyperplanes to classify instances into different classes. A hyperplane defines a decision boundary on both sides of the plane, and new instances are classified based on the side (left or right) of the hyperplane on which they fall. For example, if a new instance falls on the left hand side of the hyperplane, it will be assigned the label of the class on the left hand side. SVM can handle both linear classification and non-linear classification. Linear classification is applicable to datasets that can be separated by linear decision boundaries, and non-linear classification is applicable to datasets that cannot be classified by linear decision boundaries [27]. SVM performs non-linear classification using kernel functions [27]. Kernel functions transform a feature space into a higher dimensional feature space [27] and perform classification on the higher dimensional space. More information on SVM is presented in Section 3.1.

1.1.3 Instance selection

The ability to effectively manage large datasets is becoming a major problem due to the ever-growing rate of data worldwide. Instance selection solutions are effective tools that can be used to handle this problem. In this thesis, an instance refers to each element in a dataset. Instance selection is an important pre-processing task for data classification; it reduces storage and also improves training speed and predictive accuracy in classification problems. Instance selection techniques are generally used to remove superfluous or harmful instances from datasets [7]. Superfluous (or noisy) instances refer to instances that contribute negligibly to the decision process of a classifier, while harmful instances refer to instances that lead to high false classifications [7]. Instance selection techniques aim to select the smallest subset that will produce similar or even better predictive accuracy, compared to the entire dataset [7]. Removing or retaining too many instances can have a negative impact on classification accuracy [7], hence we must have a clear picture of the trade-off we are willing to accommodate between classification accuracy and training speed. The trade-off should be reasonable.

1.2 Problem statement

SVM is a well-known classification and regression algorithm, with good predictive accuracy and generalization capacity [28]. However, applying SVM to massive classification problems is still a major challenge [29]. SVM training time is $O(n^2)$, where n refers to the number of training instances [5]. This indicates that SVM training time increases drastically, as problem size and number of classes increase [1]. In addition, an increase in problem size will also affect the model size and storage requirements during the training and prediction stages [29]. The presence of noise in datasets also affects SVM prediction accuracy, training speed and generalization performance [29]. Also, SVM is largely affected by the quality of data used for training. Presence of noise in datasets, especially large datasets, can lead to overfitting, which will consequently affect SVM classification quality. The presence of noise can also lead to an increase in model size and consequently slows down the training and prediction stages. Obviously, these problems are major concerns, because several real-world applications require fast processing of large datasets. Hence, there is an obvious need for efficient classification algorithms that can train models within a sensible time-frame and identify new target concepts very quickly, using minimal storage space. This thesis proposes intelligent techniques for improving SVM training speed and predictive accuracy. The proposed techniques are designed to remove noisy and superfluous data, and consequently reduce n to a reasonable size.

Many classification problems can benefit from the designed techniques; however, this research focus on e-fraud detection. Several organizations communicate and perform business transactions via electronic platforms, such as emails, Internet or mobile phone calls and e-commerce. Hence, there is an obvious need to design fast and efficient systems for electronic transactions. Between October 1st, 2013 and December 1st, 2014, some companies lost a total of US$179 million to email scams. Also, seven thousand companies in the USA lost approximately US$750 to phishing, in August 2015 [30]. Unfortunately, e-fraud is on the increase. Fraudsters are devising novel and sophisticated techniques that are capable of bypassing existing e-fraud detection systems. This research undertakes to answer the following questions:

i. Can intelligent instance selection techniques improve the training speed, predictive accuracy and computational complexity of SVM?

ii. Can optimized SVM-based models improve the classification speed and predictive accuracy of e-fraud detection systems?

1.3 Research aim and objectives

The aim of this research is to design intelligent instance selection techniques for improving the training speed, predictive accuracy and computational complexity of SVM. This work also aims to apply the use of nature-inspired algorithms and machine-learning techniques to build efficient e-fraud detection techniques. The following specific objectives will be pursued to achieve the stated aim:

i. Investigate the performance of recent NI-based SVM speed optimization and instance selection algorithms.

ii. Design intelligent filter-based instance selection techniques for improving SVM training speed and storage reduction.

iii. Design intelligent wrapper-based instance selection techniques for improving SVM predictive accuracy, storage reduction and generalization performance.

iv. Implement improved SVM-based models for e-fraud detection and classification problems.

v. Evaluate the results obtained in (iv) and compare them with the standard SVM and state-of-the-art instance selection techniques.

vi. Present a statistical validation of the results obtained in (v).

1.4 Research methodology

In this research, different instance selection techniques are developed. Experiments are performed to validate the performance of these techniques using several popular test problems. Furthermore, to demonstrate the superiority of the developed techniques over standard SVM and some existing instance selection techniques, the empirical results produced by the techniques are analyzed using different statistical techniques.

1.4.1 Datasets

The proposed techniques have been validated on 24 datasets containing legitimate emails, spam emails, phishing emails, credit card fraud and twenty other problems. The spam and legitimate emails are provided by the well-known UCI ML dataset repository [19] and SpamAssassin [31], respectively. The credit card fraud dataset is provided by Kaggle Datasets [32] and the phishing emails is provided by Jose Nazario [33]. Regrettably, the phishing emails are not currently available Online; interested users are advised to contact the dataset provider. The remaining datasets are provided by UCI data repository.

1.4.2 Tools

The proposed techniques are implemented in Visual Studio 2015, using C# programming language. Different classes and methods are implemented and used in combination with the SVM methods and classes implemented in LIBSVM [34]. LIBSVM is a well-known SVM library for classification, regression and distribution estimation. Specifically, the Matthew Johnson DotNet implementation of LIBSVM is used in this research [35]. Empirical results are analyzed using the Statistical Packages for Social Sciences (SPSS).

1.5 Scope and limitation

The proposed techniques can be applied to many ML algorithms, however, this thesis focuses on SVM. Many SVM speed optimization approaches exist, including parameter optimization, feature selection and instance selection. However, instance selection proves to be one of the best [6], hence this research focuses on instance selection. While some instance selection techniques exist in the body of literature, this research focuses on nature-inspired and boundary detection algorithms. Nature-inspired algorithms can efficiently handle classification and optimization tasks [25]. They are dynamic and robust, capable of finding optimal solutions to real-world complex problems, such as e-fraud detection.

Due to confidentiality concerns, some datasets used in this research were modified by the providers. For example, the original features in the credit card fraud dataset provided by Andreas [32], were transformed to numerical features. Moreover, dataset providers did not provide sufficient information on the extracted features. Better predictive accuracy would have been achieved and better classification models would have been built if features were not transformed and if detailed information on features were provided. This would have given this researcher the liberty of extracting the desired features and building improved models.

1.6 Thesis contribution

Due to the growing world of information overload and complexities in decision-making, ML-based solutions are becoming very useful tools for many businesses. ML algorithms are known for their robustness [25], accurate data mining and classification proficiency [25, 26]. They are also known for dynamic problem solving [25]. SVM is a well-known ML algorithm that has been widely used to tackle many real-world problems, with good success. However, SVM suffers from high computational complexity, which is mainly caused by massive datasets. This research has therefore proposed and designed intelligent speed optimization techniques for improving SVM training speed, predictive accuracy and generalization performance. Specifically, the contributions of this thesis are as follows:

a. **Improvement in speed** - Some applications like video surveillance and intrusion detection require a classifier to be trained very quickly for fast classification of new target concepts. This thesis proposes seven filter-based instance selection techniques for improving SVM training speed. The filter-based techniques are divided into two categories as follows:

   Boundary detection algorithms: Two novel boundary detection algorithms are proposed in this thesis. The first algorithm (EDISA) is inspired by edge detection in image processing and the second algorithm (ACOISA) is inspired by edge selection in ACO. Both algorithms perform two functions: boundary detection and instance selection. It is noteworthy to distinguish the difference between ACOISA and other existing ACO-based instance selection techniques. In ACOISA, ACO algorithm is primarily used to identify boundaries and not to select instances. After boundary identification, K-NN is used to select instances close to the boundaries. Another novelty of ACOISA is in the heuristic value computation. ACOISA uses a novel method to compute the heuristic value for each instance.

   Nature Inspired instance selection algorithms: This thesis proposes five filter-based instance selection techniques, namely: Cuckoo Search Instance Selection Algorithm (CSISA), Bat Instance Selection Algorithm (BISA), Flower Pollination Instance Selection Algorithm (FPISA), Social Spider Instance Selection Algorithm (SSISA) and Firefly Instance Selection Algorithm (FFISA). In addition, this thesis proposes a novel fitness function for instance selection. The fitness function is utilized by the proposed filter-based techniques.

b. **Improvement in predictive accuracy** - Fast classification is often achieved at the expense of classification accuracy, and some applications, such as email classifiers, are very sensitive to slight drops in classification accuracy. Therefore, this thesis proposes five wrapper-based instance selection techniques for improving SVM predictive accuracy and training speed. The five techniques are inspired by the following NI algorithms: FPA, CSA, FFA, SSA and BA.

c. E-fraud has affected the global economy in numerous ways, hence designing optimized and improved classification models for e-fraud detection is of utmost importance. This thesis proposes improved SVM models for three popular e-fraud types: credit card fraud, spam email detection and phishing email detection. Experimental result show that EDISA (for credit card fraud), FFISA (for phishing email) and EDISA (for spam email) outperform the other proposed techniques in terms of predictive accuracy. Experimental result also show that CSISA (for credit card fraud), EDISA (for phishing email) and CSISA (for spam email) outperform the other proposed techniques in terms of speed. In addition, the robustness of the proposed techniques is further validated on 20 datasets provided by well-known UCI dataset repository. The results further show that FFISA is the most suitable for accuracy optimization, while CSISA is most suited for speed optimization.

Further applications of the proposed techniques include:

i. YouTube suggestions on music and videos, based on user historical search pattern.
ii. Shopping item suggestions on Amazon, Alibaba, eBay etc. based on user history of purchases.
iii. Real time suggestions on related paper downloads on science direct based on current paper downloaded.

All of the proposed techniques are not limited to SVM; they can be further extended to improve the performance of other ML algorithms.

1.7 Thesis outline

The remainder of this thesis is organized as follows:

Chapter 2 provides background information on different basic concepts related to this research. Moreover, Chapter 2 provides information on e-fraud detection and also presents a comprehensive survey on some existing ML-based and NI-based e-fraud detection techniques. The chapter presents a survey on some existing SVM speed optimization techniques, including feature selection techniques, instance selection techniques and parameter optimization techniques. Additionally, it highlights the limitations and strength of existing e-fraud detection techniques and also provides information on some widely-used datasets for e-fraud detection.

Chapter 3 provides an overview on SVM, together with a description of the types of SVM. The chapter provides background information on instance selection and nature-inspired algorithms. It also discusses specific details on the proposed filter-based and wrapper-based instance selection techniques. Finally, Chapter 3 describes the fitness function used by the proposed techniques and provides information on the extracted features used for classification.

Chapter 4 discusses the experimental setup and describes the measures used for evaluating the performance of the proposed techniques. The chapter also provides information on the datasets used to validate the proposed techniques. Moreover, it offers detailed experimental and statistical results produced by the proposed techniques, including discussion on the results. It concludes with a comparison of the algorithms introduced in this thesis.

Chapter 5 concludes and summarizes this thesis, and also provides recommendations for further research.

Chapter 2
Literature Review

Typically, ML algorithms can be applied to various domains, such as: text classification, video surveillance, intrusion detection, e-fraud detection and medical diagnosis. However, this thesis focuses on e-fraud detection. This section presents a comprehensive survey of existing e-fraud detection techniques. It also presents a survey of some existing speed optimization techniques.

2.1 SVM speed optimization

Some techniques have been proposed in the literature to solve SVM speed optimization problem. A sizable number of these techniques focused on speeding up SVM by reducing dataset dimension. Some studies focused on feature selection, some focused on parameter optimization and a few others focused on instance selection. The section presents a survey of some existing SVM speed optimization techniques.

2.1.1 Feature selection techniques

Many of the existing SVM optimization techniques focused on feature selection [36]. Uzer *et al.* [37] proposed a novel hybrid data classification technique. The proposed technique has two stages. The first stage focused on reducing the dimension of feature vectors. In this stage, Artificial Bee Colony (ABC) and clustering algorithm was used to select a subset of optimal features from a larger feature set. In the second stage, SVM was used to classify the selected feature subset. Using 10-fold cross validation, the algorithm was tested on some medical datasets obtained from UCI database. The test yielded positive results.

Laamari and Kamel [38] 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. The 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.

Rajalaxmia [39] solved the problem of feature selection in Type-2 diabetics using binary CSA and genetic algorithm (GA). In the study, firstly, Rajalaxmia [39] used clustering for instance selection. Next, Rajalaxmia used binary CSA and GA to select important features. Finally, the selected instances and features were used to build a model for Multilayer Perceptron (MLP) classifier. The proposed technique was evaluated and it yielded an accuracy of 99.31%.

Rodrigues *et al.* [40] proposed a feature selection approach based on BA and Optimum-Path Forest (OPF). Rodrigues *et al.* [40] used BA for feature selection, and OPF for classification. The technique was tested and it yielded promising results.

Taha *et al.* [41] proposed a feature selection approach based on BA and NB classifier. The 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.* [42] 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. The authors evaluated the approach and compared it to two other RST-based techniques, and the proposed approach outperformed both techniques.

Mousavirad and Ebrahimpour-Komleh [43] proposed a CSA-based technique for feature selection. In the study, the authors used CSA for feature extraction. Furthermore, they encoded the extracted features into a binary strings, and used them to train a K-NN classifier. The proposed approach was evaluated on five datasets obtained from the UCI data repository [19], and it yielded good results.

2.1.2 Instance selection techniques

Schölkopf *et al.* [44] combined two techniques. The first technique (also called "virtual support vector" technique) was used to improve the generalization performance of SVM, and the second technique (known as "reduced set" technique) was used to improve the classification speed of SVM. The combined technique yielded improved classification speed and generalization performance of SVM.

Guo *et al.* [45] tackled the SVM speed optimization problem by introducing a new 3-step technique. In the first step, SVM was trained to produce a number of support vectors. These support vectors were further reduced in the second step by discarding the support vectors that contributes less to the decision surface. Finally, in the last stage, SVM was trained again, using the reduced dataset. It was reported that the proposed technique yielded improved efficiency.

In another study, Lee and Olvi [46] proposed the use of a novel technique called Reduced SVM. The aim of the study was to reduce the classification speed of SVM by generating a non-linear separating surface that can be used to classify a large dataset. The non-linear separating surface was generated by firstly decomposing the entire dataset (to be classified) into smaller linear sub-problems. Afterwards, one of the sub-problems was randomly selected and used to produce the separating surface.

In a different work, Hansheng and Venu [47] proposed a new method for improving the computational speed of SVM. In the proposed method, two techniques were combined together - Principal Component Analysis (PCA) and Recursive Feature Elimination (RFE). The first technique (i.e. PCA) was used to reduce the dataset dimension, and the second technique (i.e. 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 the result revealed that it can make the computational speed of SVM faster.

Panda *et al.* [8] 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.* [8] 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. Garci *et al.* [48] proposed an instance selection technique based on EA. In the study, EA was to select generalized instances for classification in an imbalanced dataset. The technique was tested on some imbalanced datasets, and it performed better than some compared techniques. Also, in another study, Cano *et al.* [49] performed a comparative study between EA-based and non-EA based instance selection techniques. Results obtained from the study revealed that EA-based instance selection techniques yielded better data reduction rates and classification accuracy, compared to their non-EA counterpart.

Shin and Cho [50] proposed a KNN-based pattern selection technique for optimizing SVM speed. Authors designed the algorithm to select relevant training instances, based on their proximity to a decision boundary. The algorithm was tested, and it produced promising results. Angiulli and Astorino [51] proposed a SVM speed optimization algorithm, based on an existing KNN-based data reduction algorithm, previously developed by the author in [52] (called FCNN). In the study, FCNN was used in combination with SVM, to produce a faster classifier. The algorithm was tested on three large datasets, and it yielded good speed-up. Tsai and Cheng [53] proposed a clustering-based instance selection technique for bankruptcy prediction. They combined the algorithm with four classifiers: Artificial Neural Networks (ANN), decision trees, logistic regression and SVM. The technique was tested on four datasets, and its result was compared to the aforementioned four classifiers. SVM produced the best classification accuracy. Similarly, Koggalage and Halgamuge [54] proposed a clustering-based technique for improving SVM speed. In the study, the authors used K-mean clustering to select "Crisp Clusters" from a large dataset. Crisp clusters are clusters containing instances belonging to the same class. Selected crisp cluster was used as reference point for removing irrelevant instances from the training dataset. The algorithm was tested on three datasets, and it produced good data reduction rates.

Chen *et al.* [1] proposed a filter-based instance selection technique for selecting boundary instances. In the study, firstly, Chen *et al.* [1] obtained cluster centres of positive class instances using clustering algorithm. Furthermore, Chen *et al.* [1] selected boundary instances using the cluster centres as point of reference. They designed the algorithm on two postulations. Firstly, negative instances near cluster centres of a positive class are close to the boundary, and secondly, positive instances far away from cluster centres 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 contribute less to the decision surface. Also, positive instances far from a boundary and negative instances close to a boundary, contribute more to a decision surface. The authors performed some experiments to test the efficacy of the proposed technique, and the technique yielded good improvement in SVM classification speed.

Arreola *et al.* [55] proposed a decision tree based SVM classification technique. In the technique, the dataset to be classified was first disintegrated into smaller linear sub-problems. Each node in the decision tree (which consists of linear SVMs) was then used to classify the smaller problems. Furthermore, Lee and Olvi [28] extended the study of Arreola *et al.* [55]. The extended work entails decomposing the entire dataset into both linear and non-linear sub-problems.

2.1.3 Parameter optimization techniques

Temitayo *et al.* [56] proposed a GA-based technique for optimizing SVM parameters and improving SVM classification performance. The authors used GA for feature selection and parameter selection, and used SVM for classification. Authors evaluated the technique and compared it to SVM. During evaluation, authors used 4500 emails for training and 1500 emails for testing. The proposed technique yielded a classification accuracy of 93.5% within 119.562 seconds and SVM yielded a classification accuracy of 90% within 149.98 seconds.

Pereira *et al.* [57] proposed a SSA-based technique for parameter selection in SVM. Pereira *et al.* [57] used SSA to select optimal parameters suitable for SVM classification. The technique was tested, and compared to other parameter selection techniques, and it yielded better results than compared techniques. In a different study [58], the same authors proposed an SSA-based technique that performs three tasks: feature selection, parameter optimization, and a combination of feature selection and parameter optimization. The technique was tested on different public datasets, and results revealed that the combined approach yielded the best result.

Hegazy *et al.* [59] performed a comparative study on optimization of SVM parameters. In the study, Hegazy *et al.* [59] compared the performance of the following NI algorithms: FPA, BA, Modified CSA, ABC algorithm, and Particle Swarm Optimization (PSO). Hegazy *et al.* [59] tested the parameter selection capabilities of all the algorithms on LS-SVM and ANN; and all the algorithms produced good results.

Matthew and Thomas [60] proposed a technique for parameter selection based on simulated annealing. In the study, the authors used simulated annealing to improving the grid search technique used by standard SVM and consequently improve the generalization performance of SVM. The proposed algorithm was tested on two classification related problem and it yielded promising results.

Friedrichs and Igel [61] solved the problem of SVM parameter optimization using evolution strategies [62] - a main branch of EA. The authors used evolution strategies to search for the best parameter. Some experiments were performed, and results revealed that the proposed technique performed better than the exhaustive grid search technique used by SVM.

Liao and Bai [63] proposed a technique for parameter optimization and feature selection. Prior to classification, the authors first used RST to reduce the number of feature vectors to be processed. Afterwards, they used GA for parameter optimization and also used GA for feature selection. They performed some experiments to test the performance of the proposed technique and it produced improved results compared to the grid search technique.

Yi *et al.* [64] proposed a parameter optimization technique for intrusion detection (known as PSO-BP). PSO algorithm was used for parameter optimization and neural networks were used for classification. The optimized parameters were used to train the NN. The performance of PSO-BP was evaluated and the results obtained showed that the algorithm can successfully perform intrusion detection.

Saxena [65] proposed a novel method for network intrusion detection solution using the combination of PSO, K-Means and SVM. Saxena [65] used PSO for parameter optimization and K-Means to generate different training subsets. Finally, the generated training subsets were passed to SVM for classification.

2.1.4 Survey discussion: SVM optimization techniques

Table 2.1 shows a summary of all the surveyed SVM optimization techniques. As shown in the Table, many optimization techniques have been proposed in literature and most of them focused on parameter optimization, feature selection and instance selection, although some studies focused on both feature selection and parameter optimization or feature selection and instance selection. Furthermore, as shown in the Table, many optimization techniques have been explored in literature. Some authors, such as Schölkopf *et al.* [44], Hansheng and Venu [47] developed heuristics for optimizing SVM speed. Some studies used NI algorithms, including PSO, GA, simulated annealing, SSA, EA, CSA and ABC. Also, some authors used other algorithms, such as decision tree, clustering, KNN, PCA and RFE. As shown in the Table, most of the studies used clustering and KNN, and very few techniques used PCA and RFE. Also, most studies used a single algorithm for optimization and a few studies used a combination of algorithms. Among the three SVM optimization approaches, as aforementioned, instance selection is one of the most efficient [6]; however, further improvement is still required, especially in terms of speed-accuracy trade-off.

Table 2.1: Optimization techniques

Reference	**Optimization Approach**	**Algorithm**
Saxena [65]	Parameter Optimization	PSO, K-Means
Yi *et al.* [64]	Parameter Optimization	PSO
Liao and Bai [63]	Parameter Optimization and Feature Selection	GA and RST
Friedrichs and Igel [61]	Parameter optimization	Evolution Strategies
Matthew and Thomas [60]	Parameter optimization	Simulated Annealing
Pereira *et al.* [57]	Parameter optimization	SSA
Pereira *et al.* [58]	Parameter Optimization and Feature Selection	SSA
Arreola *et al.* [55]	Instance Selection	Decision Tree
Chen *et al.* [1]	Instance Selection	Clustering
Halgamuge [54]	Instance Selection	Clustering
Tsai and Cheng [53]	Instance Selection	Clustering
Angiulli and Astorino [51]	Instance Selection	KNN
Shin and Cho [50]	Instance Selection	KNN
Cano *et al.* [49]	Instance Selection	EA
Garci *et al.* [48]	Instance Selection	EA
Panda *et al.* [8]	Instance Selection	EA
Hansheng and Venu [47]	Instance Selection and Feature Selection	PCA and RFE
Lee and Olvi [46]	Instance Selection	Reduced SVM
Guo *et al.* [45]	Instance Selection	LOO
Schölkopf *et al.* [44]	Instance Selection	"virtual support vector" and "reduced set" technique
Mousavirad and Ebrahimpour-Komleh [43]	Feature Selection	CSA

Emary *et al.* [42]	Feature Selection	BA and RST
Taha *et al.* [41]	Feature Selection	BA
Rodrigues *et al.* [40]	Feature Selection	BA
Rajalaxmia [39]	Feature Selection and Instance Selection	Binary CSA, GA and Clustering
Laamari and Kamel [38]	Feature Selection	BA
Uzer *et al.* [37]	Feature Selection	ABC and Clustering

2.2 E-Fraud detection

E-fraud is a growing domain that has affected the global economy in measurable ways. Owing to e-fraud, millions of US dollars have been lost by numerous individuals and companies; thus, many researchers and companies are in search of robust and fast e-fraud detection techniques. For example, the loss incurred globally from credit and debit card transactions as at August 2013 add up to $11.27 billion [66]. In another example, in [67], the Financial Fraud Action UK (FFA UK) revealed that the loss suffered by UK card holders was summed up to £450 million, which is 16 percent higher than the loss incurred in 2012, which was £388.3 million. This section presents a survey of some techniques that have been proposed in literature to solve the problem of phishing.

2.3 Phishing

Phishing is an act that attempts to electronically obtain delicate or confidential information from users, habitually for the purpose of fraud, by creating replica website of a legitimate organization. Phishing is a classification problem and it is often perpetrated by sending deceitful and well composed emails to users. These emails usually contain links to cloned websites, and clicking on the links may re-direct users to a phishing website or a malware hosting website. Malware hosting websites are often infected with malicious codes that can gain access to private information of users and also cause damages to users' computers. Due to vast number of email messages received by various users today, separating legitimate emails from phishing emails is a challenging task. Hence the need for a quicker, robust and effective filtering technique cannot be overstated. Several approaches have been proposed in the literature, including network-based approach, blacklist, whitelist and content-based approach. Content-based approach aims at capturing the content and structural properties of a data. Blacklist contains a list of reported phishing websites, and whitelist contains a list of target companies. Network-based approaches are costly to implement, difficult to maintain and time-consuming [68]. Blacklist and whitelist based approaches yield high False Positive (FP) and False Negative (FN) rates; their effectiveness is limited to the information stored in them. This limitation makes both approaches incapable of dynamically detecting new phishing attacks as they occur. The Anti-Phishing Working Group (APWG) noted that the average uptime for a phishing website is 44.39 hours (that is, less than 2 days). Blacklist-based approaches are the most widely used approach [69]. However, content-based approaches are the most accurate and secured [70], because of their ability to discover evolving fraudulent patterns in large datasets.

Phishing can be tackled and eliminated at different levels. It can be tackled at email level [71], and at website level [72]. Dhamija *et al.* [72] proposed the use of security toolbars for web browsers. Furthermore, phishing can be tackled using visual hash. Dhamija and Tygar [73] used visual hash to identify websites that have been successfully authenticated by a browser. Buntine [74] proposed a phishing detection technique called Cryptographic identity verification. Furthermore, phishing can be reduced by training users on how to identify spoofed websites and emails.

Generally, phishing detection can be tackled from two angles: phishing website detection and phishing email detection [75]. Table 2.2 shows a summary of the phishing detection techniques reviewed in this paper. As shown in the Table, NI algorithms and Model-based techniques have not been fully explored. Many studies focused on heuristic, rule-based techniques and ML-based techniques. As shown in the review, ML-based techniques yielded better results. Some authors combined NI and ML algorithms. NI algorithms were mostly used for feature selection. Also, some studies introduced classifier ensemble techniques, which involved combining outputs of different classifier. Additionally, some authors proposed new features for phishing detection based on Uniform Resource Locator (URL) properties, structure of email, content of email, heuristics and external sources. Additionally, as shown in Table 2.2, some studies used blacklist and whitelist. However, the blacklist-based detection technique requires regular update; hence it is static [75]. These techniques cannot effectively detect zero hour phishing attacks [76], and require more human resources [77]. These limitations stimulated the need for an improved and dynamic phishing detection technique.

Table 2.2: Summary of phishing detection techniques

Technique Category	**Name of Technique**	**Reference(s)**
NI	ACO	[78]
	AIS	[79]
	PSO	[78]
	Neuro-Fuzzy Logic	[80]
ML	SVM	[81-84]
	ANN	[85]
	Naïve Bayesian theorem	[86]
	BN	[87]
	RF	[86, 88-91]
	logistic regression	[92]
	C4.5, Ripper, PART, PRISM and CBA	[93, 94]
	Decision tree	[86, 95]
	LDA	[91]
Blacklist-based and Whitelist based	Blacklistbased and Whitelist Based	[96-98]
Heuristic and Rule Based	Heuristic and Rule Based	[77, 99-104]
Ensemble Based	Ensemble Based	[105-107]
Model-Based	Dynamic Markov chain and Latent Class-Topic	[75]

Several techniques have been proposed in literature to tackle the phishing problem. Some of the techniques are based on ML algorithms, while some are designed using an ensemble of different classifiers. Moreover, some of the techniques are based on heuristics and rules, while some are based on NI algorithms. This section provides a survey of some existing phishing email detection techniques.

2.3.1 Machine learning based phishing email detection technique

Chandrasekaran *et al.* [83] developed a SVM-based technique for phishing email detection. The authors extracted 25 features and used them to build a SVM model. They evaluated the technique on a dataset consisting of 200 phishing emails and 200 legitimate emails (obtained from news groups, bulletin boards and personal inbox of users). During evaluation, they extracted 23 style marker features and two structural features. The style marker features consists of the following: total number of characters, total number of unique words, vocabulary richness, frequency distribution of 18 keywords and total number of functional words. The structural feature includes structure of email subject line and structure of greeting provided in email body. The results of the evaluation revealed that the technique produced a classification accuracy of 88% over five runs.

Fette *et al.* [89] developed a RF-based classifier for phishing email detection (called PILFER). In the study, the authors proposed 10 features. Eight were extracted from emails, and two were extracted from external sources (SpamAssassin, an existing anti-spam filter) and WHOIS server). The authors evaluated the technique on a dataset containing 860 phishing emails and 6950 non-phishing email and it yielded a classification accuracy of 99.5% and an FP rate of 0.0013%.

Bergholz *et al.* [75] introduced two model-based features based on Dynamic Markov chain and Latent Class-Topic for phishing email detection. The authors developed the one Dynamic Markov chain model for phishing and spam email and the one Latent Class-Topic model for phishing email. Furthermore, they trained the Dynamic Markov chain model and used its output to generate four features. Two of the features are based on email class likelihood, and the other two are based on email membership. Moreover, the authors extracted words that appear together in emails and used them to train the Latent Class-Topic Model. Afterwards, they combined the outputs from the model-based features with 27 other basic features, and used them to train a SVM classifier. Finally, they evaluated the performance of the technique on four different datasets and it yielded an overall classification accuracy of 99.85%, FP rate of 0.01% and FN rate of 1.30%.

Amin *et al.* [90] designed a RF-based solution for classification of phishing email targeted at single users or small group of users. The authors referred to such emails as Targeted Malicious Email (TME). Current conventional techniques are designed to detect emails sent to vast volume of users [90]. In the study, the authors developed a specialized filtering technique for TMEs by firstly extracting word-based features from emails. Afterwards, they assigned weights to each extracted feature (using TF-IDF algorithm) and selected features whose weights is above a pre-defined threshold. Selected features were then used for classification. The authors evaluated the performance of the proposed technique on three datasets collected from an unnamed company. The first, second and third datasets respectively contained 2,315 TMEs, 20,894 Non Targeted Malicious Emails (NTME) and a combination of TME and NTME. The technique yielded positive results.

Liping *et al.* [95] designed a decision tree parser (or a decision tree translation system) for converting decision trees into an implementable program language. The translation system consists of four components: feature generator, learner, inductor and a classifier. During classification, the feature generator converts extracted features into vectors and passes them to the learner module for training. Afterwards, Information Gain (IG) for all the features was generated and passed to the inductor. Furthermore, features with high IG were selected and used re-train the classifier. The training cycle continues until the best set of features is identified. The best features are used to construct the final classifier. The authors evaluated the system on a dataset containing 613,048 legitimate emails and 46,525 phishing emails and it yielded a classification accuracy of 99.5%.

Sanchez *et al.* [84] developed a SVM-based classifier for classification of banking-based emails. In the study, the authors developed three rules. The first rule handles emails containing account for email service provider. The second rule detects inconsistencies in senders' geographical locations. This rule is triggered if origin of message is not consistent with the bank's location. The third rule checks whether the mail server (i.e. where the message originates from) is an authorized server for the bank. A message is considered to be legitimate if it passes all the three rules. The authors evaluated the performance of the technique and it yielded a classification accuracy of 98.7%.

Toolan and Carthy [94] ptheroposed a feature selection technique for phishing and spam email detection. The authors' primary aim was to show the effectiveness of IG for feature selection and to provide a feature set for building an effective classifier. In the study, they extracted 40 features and computed their IG and entropy. Thereafter, they selected features with high IG and evaluated them using C4.5 algorithm. They tested the technique on three datasets. The first dataset (containing 4202 ham and 1895 spam email) was used to evaluate importance of features in spam detection. The second dataset (containing 4563 phishing email) was used to evaluate feature importance in phishing detection. The third dataset (containing the ham, spam and phishing corpora) was used to evaluate importance of features in a real life system. A real life system typically contains a combination of spam, ham and phishing emails. Results from the experiments revealed that classifiers trained with features having high IG will perform better compared to classifiers trained with lowly ranked IG features.

Xing *et al.* [79] proposed an AIS based solution for phishing email detection. During implementation, the authors extracted features from emails and used them to generate memory detectors for email classification. A memory detector contains the email address of phisher, a set of phishing tokens and number of links in an email. Furthermore, the authors saved a copy of the memory detectors and performed mutation on the original copies to form immature detectors. Afterwards, they converted the immature detectors to matured detectors using the process of negative selection. During mutation, new factors were reproduced and used to replace old values in the memory detectors. The mutated matured detectors were then combined with memory detectors and used to classify new emails. Thereafter, the authors assigned weight to each detector and computed a final score using a formula defined in [79]. If the score is greater than a user defined threshold, the detector is treated as fired. An alarm is raised if number of fired detectors is greater than a user defined value (called fire alarm range). A fired detector will be saved as memory detector for future classification. Additionally, matured detectors that do not fire will be eliminated after a period of time. The authors performed some experiments on a dataset containing 100 phishing emails and 400 ham emails, and it yielded promising results.

Debarr *et al.* [91] proposed a phishing email detection technique capable of providing solution to spear phishing (phishing email sent to known email contacts). In the study, the authors used spectral clustering, Latent Dirichlet Allocation (LDA) and RF for classification. Furthermore, they extracted tokens from subject and body section of each email and used LDA to randomly select tokens for removal. They then extracted URLs from emails and passed them to Spectral Clustering for reduction. Afterwards, the authors used the reduced emails to build a RF-based classifier. They evaluated the proposed technique on a dataset consisting of 4150 ham emails and 4559 phishing emails. During experiments, the authors constructed RF classifier for both LDA and Spectral clustering. Results revealed that Spectral clustering and LDA are good phishing detection techniques.

Marchal *et al.* [108] proposed a phishing URL detection technique called PhishStorm. The primary aim of the study was to identify registered domain names different from targeted brands. The authors extracted 12 features based on intra-URL relatedness and URL popularity. Intra-URL relatedness refers to similarities between components of a URL, and it was obtained by using Bloom Filter [109] – a statistical data structure tool. During classification, the authors extracted URLs from email, removed words from the URL and passed them to a search engine (Google and Yahoo). Afterwards, they used the results obtained from search engine to compose a word set, based on inter-URL relatedness and URL popularity. They then computed IG for each word set and passed the word sets with high IG to seven ML algorithm for classification. They evaluated the technique and a classification accuracy of 94.91% was obtained.

2.3.2 Ensemble based phishing email detection technique

Toolan and Carthy [106] proposed a phishing email detection technique based on classifier ensemble of four classifiers: KNN, SVM, NB and Linear regression. The classification stage was divided into two. C5.0 was used to perform the first classification. Thereafter, emails classified as legitimate were passed to the classifier ensemble for re-classification. The authors evaluated the technique on a dataset containing 4116 phishing emails and 4202 non-phishing emails. During evaluation, the authors firstly evaluated the performances of the individual classifiers (i.e. C5.0, KNN, SVM, NB and Linear Regression). Afterwards, they used the best four performing classifiers to build four different classifier ensembles, consisting of three classifiers each. After evaluation of the ensembles, C5.0 and SVM performed better in terms of classification accuracy. However, the ensemble classifier performed better in terms of recall. Consequently, the authors combined the best individual classifier (i.e. C5.0) with the best ensemble classifier. The goal of combining both classifiers was to produce a robust classifier with good recall and high classification accuracy. The combined technique yielded an average classification accuracy of 99.31%.

Saberi *et al.* [107] proposed an ensemble based solution for phishing email detection. In the study, the authors worked with three classifiers: KNN, Poisson probabilistic theory and Bayesian probability theory. The authors trained the three classifiers and combined their results using ensemble approach (majority voting). The approach was evaluated on a dataset containing 4500 scams, 1500 legitimate emails and 529 phishing emails and a classification accuracy of 94.4% and FP rate of 0.08% was achieved.

2.3.3 Nature inspired-based phishing email detection techniques

Radha and Valarmathi [78] developed an hybridized fuzzy-based phishing solution for phishing e-banking website. The solution consists of three algorithms: Association and classification data mining algorithm, ACO and PSO. During implementation, the authors extracted features and assigned a fuzzy membership value to each of them. Afterwards, they generated "if then" rules with the data mining algorithm and performed aggregation on all the rules that exceeded a minimum confidence value. Moreover, the authors de-fuzzified the fuzzy set and optimized the rules using ACO and PSO. Finally, they used the optimized rules to classify new websites. They performed a series of experiments on a dataset containing 1052 URLs and reported that the technique yielded an accuracy of 91%.

2.3.4 Survey discussion: Phishing detection techniques

Based on the surveyed studies, it is apparent that various techniques have been proposed to solve the problem of phishing website and phishing email detection. Most authors focused on phishing website detection. Most studies focused on URL-based features. Garera *et al.* [96] introduced seven URL-based phishing features. Also, techniques proposed by authors in [81], [77] and [88] are URL based. Yearwood *et al.* [82] noted that URLs are the most important features used by phishers, because it is URLs that redirect users to spoofed website. If spoofed URLs can be detected and prevented from re-directing to spoofed websites, phishing attacks will be reduced drastically. Furthermore, Bergholz *et al.* [75] introduced a novel phishing email detection technique based on two models: Dynamic Marcov Chain and Latent Class-topic models. The technique yielded excellent results; however, it is time-consuming. Extra time is required for training the two models. Additionally, authors noted that headers and attachment were removed from emails before classification was performed. Meanwhile, headers and attachments are good phishing indicators [75].

Other studies proposed heuristics-based solutions. Garera *et al.* [96] proposed a heuristic, based on Google index infrastructure. Also, Zhang *et al.* [77] proposed a heuristic, based on TF-IDF and Robust Hyperlink Algorithm. However, heuristics are not reliable phishing indicators and their effectiveness is limited. Network failure, slow network speed, late response and wrong feedback from external source can drastically affect classification performance of heuristic-based classifiers. Additionally, heuristic-based techniques are not very efficient, because phishers can design attacks that can bypass heuristics [77]. Some authors proposed blacklist and whitelist based approaches. Garera *et al.* [96] created a blacklist and whitelist of phishing and non-phishing URLs respectively. Likarish *et al.* [110] also used whitelist. Blacklist- and whitelist-based techniques are not reliable; they cannot effectively detect zero-day attacks.

Some studies worked on word-based features, such as [106] and [83]. Unlike external features that depends on extra resources (such as network connectivity, bandwidth, external source availability), word-based features are good phishing indicators: they can be easily extracted from emails and web pages. Furthermore, some authors proposed rule-based solutions. Authors in [85], [99], [111], [84], [104] and [78] used such techniques. However, rule-based systems require regular update; hence, they cannot effectively detect zero-day attacks. They can be easily bypassed by phishers. Aggarwal *et al.* [86] developed a phishing detection technique for Tweeter. In the study, authors used a combination of different features including features extracted from tweeter. Authors also designed an extension for Chrome browser. In other studies, authors in [102] and [103] proposed feature selection techniques for phishing detection. Authors in [102] focused on URL-based features, and authors in [103] focused on content and behaviour-based features. Different ML and NI techniques have been proposed. Authors in [75], [81] and [83] applied such techniques.

Many of the proposed ML-based techniques focused on the SVM algorithm. Authors in [75], [84] and [83] used SVM for phishing email classification. Huang *et al.* [81] developed an improved SVM-based technique for phishing URL detection. However, one of the major drawbacks of SVM is speed. The training time for SVM is estimated to be $O(n)^2$ [8, 112]. Other ML and NI based algorithms explored in the literature include logistic regression, NN, Bayesian algorithm, RF, PSO, ensemble classifiers. Authors in [96] and [92] developed a logistic regression based model for phishing URL detection. Both models were tested on millions of URLs and they yielded positive results. Furthermore, Martin *et al.* [85] developed a NN-based framework for phishing website detection. Likarish *et al.* [110] developed a phishing detection solution based on Bayesian algorithm. The authors noted that the solution is the first Bayesian based phishing website detection technique. However, the technique was evaluated on few data instances (120 websites); hence its effectiveness is not guaranteed. The effectiveness of a Bayesian-based classifier depends on the number of instances used to train it [110]. Moreover, authors noted that there is little delay during page load, implying that the technique can be slow if large volume of dataset is processed. Fette *et al.* [89] developed a RF-based phishing email classification technique, which yielded good results. In the study, the authors introduced a robust set of internal and external features extracted from email content and from external sources (WHOIS servers). However, the technique's performance depends on network speed and correctness of information received from external source. Slow network leads to delayed classification, and incorrect information leads to increased FP rates and reduced classification accuracy. The authors reported that WHOIS servers may return result in non-standardized format, making it difficult to process. Also, not all domain names will be present in the server, especially domain names that have been blacklisted and removed.

Deshmukh *et al.* [113] proposed a specialized phishing email detection technique for classification of emails targeted at single users or small groups of users. Although the technique yielded promising results, authors did not provide details of dataset used for evaluation. In a different study, Toolan and Carthy [106] developed a phishing email classification technique based on classifier ensembles. In the study, the authors introduced a robust technique (called R-Boost) that combines the classification strength of C5.0 and high recall of the classifier ensemble. The technique yielded good results; however, it is time- consuming and complex. It involves classification in two folds: classification for C5.0 and classification for the ensemble. Similarly, authors in [105, 106, 114] proposed ensemble based phishing detection solutions. However, ensemble classification is time-consuming, because it involves running three classifiers for classification of one data instance. Xing *et al.* [79] proposed an AIS based solution for phishing email detection. Although the solution yielded promising result, the threshold value assigned to the system is static. The authors noted that the system can be improved by introducing dynamic fire-threshold value and Fire-Alarm-Range value to the system. AIS have not been fully explored in the domain of phishing detection.

2.3.5 Limitations of phishing email detection techniques

Many phishing email detection techniques have been proposed in literature. Some of these techniques achieved remarkable results, while some produced poor results. Some techniques require installation of infrastructure, which some email clients do not have [115]. For example, S/MIME and PGP (standards for signing email digitally) require the installation of an infrastructure that supports digital signing and verification [115]. Some of the existing techniques are static [75], hence they cannot effectively handle new or emerging attacks [76]. Additionally, some of the existing techniques, such as rule-based techniques, can be easily bypassed by phishers. Some of the proposed techniques are slow, due to inputs from external sources or processing of large features, such as images. Future work should focus on designing simple, fast and dynamic classification models, capable of accurately tacking existing and emerging phishing attacks.

2.4 Spam email

Spam refers to unwanted emails received by users having no current relationship with the sender. Spam is a worldwide problem that has affected the globe enormously. Generally, spam email messages aims to advertise pornographic websites, products, or perpetrate fraud [116]. Email addresses used for spamming are collected from different sources including websites, chat rooms, newsgroups etc. [117]. Since the emergence of information and communication technology, communication via email became prevalent [118]. Good percentage of organizations and individuals worldwide utilize email as a major means of communication. In 2016, email accounts worldwide were estimated to be over 4.3 billion [119]. Undoubtedly, email is one of the fastest, cheapest and most convenient means of communication [120]. However, approximately 92% of received emails are spam [121]. Billions of emails received by ISPs in recent times are spam [122]. In 2006, about 12.4 billion of 31 billion emails sent per day were considered to be spam [123]. On average, an email owner receives between 10-50 spam emails daily [124]. In 2006, Message Labs [125] reported that spam accounted for about 58% of network traffic. Network traffic in turn caused a delay in email delivery. Moreover, spam cost different service providers and organization a bandwidth loss of millions and billions of dollars [116]. It also cost them loss in employee productivity. Spam emails leads to wastage of bandwidth, wastage of time, wastage of resources, wastage of storage space, and wastage of money [120]. Additionally, it exposes users to unpleasant content, and provides means for phishing attacks and distribution of malicious software like Trojan and worms [126].

This section presents a survey on some existing spam email detection techniques. Specifically, this survey is centred on NI and ML based spam email detection techniques. Table 2.3 presents a list of all the spam email detection techniques surveyed in this research. The Table reveals that GA is one of the most popular NI techniques that has been used in literature. Although, GA requires more parameter tuning [127], it is a good optimization technique suitable for optimal subset selection [56]. AIS is another NI technique that has been used to handle spam email detection in literature; however, as shown in Table 2.3, it has not been fully explored. Other NI techniques that have been used to tackle spam email detection include ACO, EA, PSO, FFA, BPSO, SAIS. ML techniques have also been used to tackle spam email detection. Table 2.3 shows that NB is the one of the most popular ML techniques that has been used in literature. NB algorithm is easy to implement; it is a good algorithm that can used in combination with other algorithms to build an improved spam email system [128]. However, NB must be trained on a large volume of dataset for better performance [129]. Other ML techniques that have been explored include SVM, ANN, RVM, RST, decision tree and C4.5. Additionally, some authors developed hybridized spam email detection techniques. This section presents a survey of some existing spam email detection techniques, and also outlines their various limitations and strength.

2.4.1 Machine learning based spam email detection techniques

Spam email has been a major problem for many decades, hence different techniques have been developed in literature to handle this problem. This section presents a survey of some ML-based spam email detection techniques.

2.4.1.1 SVM based techniques

Tseng and Ming-Syan [130] designed an improved spam detection system (called MailNet), capable of adjusting to different networks. In the study, the authors constructed an email network consisting of different users, represented as nodes. They extracted features from pure nodes. Pure nodes refer to nodes that have sent out either spam email or legitimate email, but not both. The authors noted that if the number of nodes is above a specified threshold, it will be reduced further.

Table 2.3: Summary of existing spam email detection techniques

Technique Category	**Name of Technique**	**Reference(s)**
NI	ACO	[126, 131]
	GA	[122, 129, 132-134]
	EA	[135, 136]
	AIS	[137]
	PSO	[138, 139]
ML	SVM	[118, 140]
	ANN	[116, 141, 142]
	Naïve Bayesian theorem	[120, 124, 128, 143, 144]
Hybridized Techniques	GA and SVM	[56]
	Taguchi method and Staelin method	[127]
	NB, Relevance Vector Machine, SVM and Neural Network.	[118]
	ANN And GA	[145]
	ACO and SVM	[117]
	Bayesian and NN	[146]
	LDA and ACO	[147]
	ACO, rough set and GA	[148]
	GA, NN, AIS	[149]
	Firefly and NB	[150]
	Binary PSO (BPSO), decision tree, C4.5 algorithm	[121]
	Rough Sets and PSO	[151]
	Simple AIS and PSO	[152]

Additionally, after feature extraction, the authors normalized the extracted feature vectors, and used them to train SVM. The trained SVM was then used to classify incoming emails. The system performs incremental update periodically. During updates, when new emails enters the network, the corresponding node is updated. The updated nodes, alongside support vectors from previous stages, are used to re-train the model. The authors performed some experiments to evaluate the performance of the proposed technique. During the experiment, the authors extracted seven features from a dataset consisting of 2,136,329 ham emails and 729,304 spam emails. MailNet yielded a True Positive (TP) rate between the ranges of 94.06% to 95.38%, and FP rate between the ranges of 1.21% and 1.75%.

Xiao-li *et al.* [153] proposed a spam email detection technique capable of reducing SVM misclassification rate. The authors noted that the technique is biased towards legitimate emails. They introduced a slack variable ($S_i > 0$) that shows the importance of each class, and to reduce misclassification rate of a given class. Slack variables for all samples in dataset were multiplied by weight of the sample (sample weight) and weight of class (class weight). Legitimate emails are assigned higher weights, hence their misclassification rate is reduced. The authors evaluated performance of the proposed techniques by performing different set of experiments using different class weights. RBF kernel was used in all the experiments. The results obtained from the experiments indicated that weighted SVM approach can control classification performance of SVM. The first group of experiments (containing a class weight of $c+ = 2$ and $c- = 1$) yielded a recall, precision and classification accuracy of 96.5%, 97.47% and 97.00% respectively. The second group of experiments (containing a class weight of $c+ = 5$ and $c- = 1$) yielded a recall, precision and classification accuracy of 93.00%, 98.41% and 95.45% respectively. The third group of experiments (containing a class weight of $c+ = 10$ and $c- = 1$) yielded a recall, precision and classification accuracy of 85.50%, 99.44% and 94.50% respectively.

2.4.1.2 ANN based techniques

Nosseir *et al.* [141] proposed a novel spam email detection approach based on characters and words. In the study, the authors extracted email from dataset, and pre-processed the emails. During pre-processing, they removed stop words and other form of noise (such as misspelt words) from the extracted emails. Also, they stemmed the email content and divided them into three groups, based on length of words. Words with three, four and five characters respectively were placed in three different groups. The authors divided the words into two classes (bad and good), based on their meaning. Additionally, they classified the words into different categories, such as advertisement, financial, etc. Each category was assigned a given weight, which can be adjusted by the user. Moreover, the words in each category were then passed to three different networks for training. The first, second and third NN have three layers with three, four and five input neurons respectively and two output neurons. Finally, the authors tested the trained network on incoming emails. If an email does not contain any of the 'bad' words, it is automatically classified as ham; otherwise, it will be classified based on the number of bad words identified in the email. The authors calculated the number of each bad word and multiplied the number by the weight of its category. The calculated weight is used by a decision function to classify the email as ham or spam. The authors designed three networks, each designed to handle three, four and five characters respectively. Each network was trained and tested. The test was based on three different datasets, containing three, four and five character words respectively. Each dataset contains 20 good words and 20 bad words. The technique yielded the following Types I and Type II FP rate for the three, four and five character NN: 0.131364 and 0.999962, 0.0003 and 0.7953, 0.0015 and 0.9990.

Wu and Tsai [142] proposed a behaviour-based spam email detection technique. The authors extracted the behaviour-based features from email header and email syslogs, unlike keyword-based features that are extracted from body of emails. Syslogs refer to record files that are added to auditing files when a Mail Transfer agent delivers an email [142]. Syslogs contain a description of the email delivery [142]. The authors collected over 10,000 spam emails and 20,000 ham emails and extracted features from their email headers and syslogs. During the feature extraction, they analyzed the email headers and selected the fields that most frequently occur. Also, they analyzed some email syslogs and selected fields that occurred most frequently. They then selected six header feature and four syslog feature for training. They noted that some fields in syslogs and email headers are related and should be the same for each email. Based on this assertion, they extracted sixteen more features. Finally, the authors combined all the extracted features (referred to as behaviour-based feature), assigned values to them (based on some heuristics explained in [142]) and used them to train an NN, consisting of 26 input nodes, two hidden layer and one output layer. The trained network was then tested and modified if the output is not satisfactory. Afterwards, the final model was used to classify incoming emails. The authors performed several experiments to test the robustness of proposed technique. The technique was evaluated on a dataset containing

43,890 emails. 21, 945 spam and 21,956 ham emails. The best classification accuracy obtained from the experiments is 99.75%.

2.4.1.3 Naïve bayesian based techniques

Kufandirimbwa and Gotora [116] designed a spam filtering technique based on ANN and Perceptron Learning Rule (PLR). The authors extracted features from header and body part of emails and used them for training. They used the gradient method for training. During training, true gradient was evaluated on a single data instance, and gradient weight was adjusted gradually until a pre-defined stopping condition was reached. For each iteration, the authors computed an error and weight adjustment value, and used the values to adjust the actual weight value of email. After adjustment, emails with the adjusted weight were selected and parsed as input to the neural network, which then computes an output. If the computed and expected output are not equal, the weight will be re-adjusted, and a message will be parsed again to the network. This process is repeated until the network generates an output that is equal to the expected output or until the maximum number of iterations is met. The authors performed different experiments using a dataset consisting of 140 emails. The proposed technique produced a FP rate of 97.14%.

Savita and Santoshkumar [120] proposed a spam email detection technique based on naïve Bayesian theorem. The authors extracted keywords from emails in dataset, and calculated spam scores for each of them. Afterwards, each keyword, and their corresponding spam score were saved in a database and used for classification of incoming emails. An incoming email is classified as spam or ham based on its spam score. If its spam score matches a predefined spam probability, it is classified as spam; otherwise, it is classified as ham. The authors performed some experiments on a dataset consisting of 12600 emails, and the technique produced a classification accuracy of 95%.

Bhagyashri *et al.* [124] developed a spam email filtering system based on BN. In the study, the authors calculated TF for words in each email, and trained Bayesian classifier with words having high TF. The trained network was then used to classify incoming emails. During classification, incoming emails were split into tokens and spam score for each token was calculated and summed. If the total probability is greater than 0.5, the email is labeled as spam; otherwise, it is labeled as ham. The authors evaluated the technique on a dataset containing 50 ham emails and 50 spam emails, and it yielded a classification accuracy of 90%.

Vira *et al.* [143] proposed a spam email classification technique based on Bayesian theorem. The authors trained classifiers on word-based features extracted from emails. Prior to training, they extracted features from dataset, grouped them into categories and saved them in a database. Classification of incoming emails is performed based on their conditional probability and pre-defined threshold saved in database. The authors evaluated the technique on 5175 emails and it produced a classification accuracy of 96.7%.

Issac *et al.* [154] designed a keyword-based BN technique for spam email detection. The authors designed three techniques using single keywords, multiple keywords and context matching keywords. In the first technique, they extracted single and multiple keywords from email and assigned weights to the keywords. Afterwards, a Bayesian score for all the keywords were calculated (using a formula described in [154]) and totalled. The total score gives the score for an email. The second technique is similar to the first technique. However, in the second technique, weights are assigned to multiple keywords. A spam score for each email was calculated using a formula described in [154]. In the third technique, a context score for keywords was added. For every keyword in the dataset, a matching context score was calculated. The score is calculated based on number of times each keyword appears in a dataset. The authors evaluated the three proposed techniques on a dataset containing 2412 ham emails and 481 spam emails. It yielded an average FP rate of 7.86%, 5.03% and 4.83% respectively, and an average False Negative (FN) rate of 21.50%, 14.94% and 12.78% respectively.

2.4.2 Nature inspired spam email detection techniques

Table 2.3 shows a summary of some existing spam email detection techniques. As shown in the table, NI-based techniques are valuable techniques suitable for improving the performance of spam email filtering systems. They are mostly used in combination with other techniques. This section presents a literature review of some recent NI-based techniques proposed in the literature.

2.4.2.1 Ant colony optimization based techniques

El-Alfy [126] proposed an ACO-based anti-spam system. Prior to training, the authors extracted features from emails, pre-processed and saved them in a database. Furthermore, they extracted capitalized words, special characters and punctuations, and saved them in a database. Additionally, they calculated IG for all the saved features and used ACO to generate classification rules (for incoming emails) with features having high IG. Authors evaluated the system on three datasets, each containing 4601 instances. Evaluation yielded a classification accuracy of 90.29%.

Liu *et al.* [131] proposed an ACO-based approach for feature selection. In the study, the authors designed two fuzzy-based controllers for regulating ACO parameters adaptively, namely, pheromone evaporation rate and number of ants. The authors noted that the proposed fuzzy controllers can improve the performance of a spam filter by balancing between search space exploration and exploitation. During the experiment, they initialized the parameters of ACO, and set number of ants and number of iteration to specific values. Furthermore, they randomly generated an ant population according to the number of ants specified. They randomly selected and initialized feature subsets for each ant, and used IG to select relevance of each feature. The authors used ACO to further optimize the feature subset. During ACO optimization, for each iteration, the authors passed the optimized feature subset to SVM for evaluation. At the end of each iteration, the pheromones, number of ants and pheromone evaporation rate was updated using a fuzzy controller. The process is repeated until a pre-defined number of iteration is reached. The authors evaluated the proposed algorithm on ten datasets and compared it to standard ACO algorithm, standard PSO algorithm and standard GA. Results from the evaluation showed that the proposed technique outperformed the other methods and produced a classification accuracy of 98.2%.

2.4.2.2 Genetic algorithm based techniques

Shrivastava and Bindu [132] proposed a GA-based spam email detection technique. The authors extracted features from body section of emails and used GA to generate classification rules for incoming emails. The proposed technique was evaluated on 500 emails (300 hams and 200 spams) and it yielded a classification accuracy of 82%. The same authors in [134] proposed another email detection technique using GA and a Heuristic based function. In the study, they extracted word-based features and matched them to a database of spam words. Furthermore, spam scores were calculated for words that exist in the database, and encoded into binary chromosomes. Afterwards, the authors used GA to perform classification. During classification, incoming emails were processed, and encoded as chromosomes. Moreover, crossover and mutation were also performed. The technique was evaluated and it yielded a classification accuracy of 82%.

Behjat *et al.* [133] proposed a GA-based technique for spam email detection. Firstly, the authors extracted features from dataset and calculated their respective TF. Based on the calculated term frequencies, they used GA to select optimal features. They passed the selected features to a Multiple Layer Perceptron (MLP) NN for training. Finally, they tested the trained network on new emails from test set. The incoming emails were first pre-processed and then passed to the network for testing. The test produced a classification accuracy of 99.68%. Performance of technique was compared to other ML techniques and it performed better.

In an Honours thesis, James [129] investigated the performance of GA, Memetic algorithm and Multi-objective GA in solving problem of assigning correct weights to various tests performed by spam filters. The thesis objective was to optimize the performance of spam filters. Specifically, the author focused on optimizing the performance of SpamAssasin [155], a popular spam email filter. In the study, the author designed and implemented three NI-based techniques using GA, MA and multi-objective GA. In the GA-based technique, firstly, he initialized GA parameters and also set a selection level, which was used to determine the total number of selected solution. Moreover, he randomly generated a score for each test performed by SpamAssassin. He performed recombination and mutation, and calculated the fitness value for each population, then selected population with the best fitness function and passed it to the next level. He repeated the process until selection level was reached. He noted that all parameters used in the study are the same, with the exception of fitness function and ranking method. Multi-objective GA has two fitness functions. The first measures classification accuracy of the classifier, and the second measures misclassification rates. Also, to rank solutions, the multi-objective GA used a method introduced in NSGAII algorithm (discussed in [129]). James [129] noted that the proposed MA is similar to GA. The major difference is local search. He used Hill-climbing local search. In the local search, firstly, a local optimum for each score (assigned to each test) was set. The score was increased and a new solution was determined. If the new score improves the solution performance, then the score is retained, else it is discarded. This process is repeated until no improvement in performance is observed. The author performed several experiments on datasets obtained from TREC 2005 dataset and the three algorithms produced promising results.

Sorayya and Seyed [122] proposed a GA-based feature selection technique (called GAFS). The authors used GA for feature selection. During feature selection, they converted extracted email words into a string of chromosomes, where each string in the chromosome is a binary number representing each feature. Afterwards, they randomly generated the initial population and selected all string entries that are equal to 1 for classification. Furthermore, they performed crossover and mutation on each of the selected chromosomes and calculated their respective fitness value using F-Score. Chromosomes with F-Score greater than a pre-defined threshold was selected. Finally, the selected chromosomes were passed to KNN and BN for classification. Some experiments were performed to investigate the performance of the proposed technique. During experiments, the authors evaluated KNN and BN using GAFS feature selection technique, and it yielded promising results.

2.4.2.3 Evolutionary algorithm based techniques

James *et al.* [135] proposed an EA based technique to improve the performance of SpamAssassin. In the study, the authors used EA for weight optimization. They used EA to optimize some set of weights used by SpamAssassin for classification. They performed some experiments using a population size of 200 and a dataset containing 52,790 spam emails and 39,399 ham emails. In the experiments, 90% of the dataset was used for training, and 10% was used for testing. The experiments yielded a classification accuracy of 94%.

Cortez *et al.* [136] proposed an EA-based feature selection technique for spam email filtering. In the study, the authors extracted features from dataset and pre-processed them. During pre-processing, they removed non-numeric characters and HTML tags. They also removed words less than, or equal to two characters, and further reduced the dimensionality of feature space, by removing words with low TF. Furthermore, they calculated IG for the remaining words and selected words with high IG. Finally, they used EA to train NB classifier. The authors evaluated the technique on a dataset containing 19,196 emails and it yielded an accuracy of 97%.

2.4.2.4 AIS based techniques

Abi-Haidar and Rocha [137] presented a solution for spam email detection based on cross-regulation model of AIS, called ICRM. In the study, the authors trained AIS with features extracted and randomly selected from dataset. They tested the solution on six datasets, each containing 1000 ham and spam emails. They also compared the performance to NB and VTT (a binary classification algorithm). The proposed technique yielded an average classification accuracy of 89%.

2.4.2.5 PSO based techniques

Prilepok *et al.* [138] proposed two algorithms for spam email detection, based on PSO and NB. The authors hybridized the Bayesian based filter with a data compression algorithm. Afterwards, they used PSO for feature selection and NB for classification. They noted that feature selection was performed using Graphical Processing Unit (GPU) units. In GPU units, tasks are executed in parallel, to improve processing speed. The authors tested the algorithms on a dataset containing 48,360 spam emails and 36,450 ham emails obtained from TREC. The PSO-based filter produced a precision of over 60% for ham emails and 50% for spam emails. Also, the Bayesian based filter, yielded a precision of 99% for ham mails and 90% for spam mail was achieved.

2.4.3 Hybridized spam email detection techniques

Hsu and Yu [127] used the Taguchi method in combination with Staelin method to develop an SVM-based solution for spam email classification. The authors used the Staelin method for parameter optimization and SVM for classification. They performed some experiments to evaluate the performance of the hybridized technique. During the experiments, they randomly selected 500 ham emails and 500 spam emails from six different datasets, and used them for training, and the proposed system yielded a classification accuracy of 99.60%.

Yu and Xu [118] performed a comparative study of the performance of four ML algorithms on spam email classification, namely, NB, RVM, SVM and NN. The authors trained different classifiers on features extracted from email. They performed some experiments to evaluate the performance of the four algorithms. The results from experiments revealed the following:

i. NN is not a suitable stand-alone spam email filter. This is because NN is sensitive to changes in training set and NN can be over-fitted by training set.
ii. Performance of SVM and RVM is better than NB. Furthermore, performance of RVM and SVM is not influenced by the entire data; it is influenced by the support vectors or relevance vectors (in the case of RVM).
iii. Number of features used has slight effect on RVM and SVM but not on NB and NN. Moreover, RVM and SVM perform better when trained with larger number of features.
iv. RVM and SVM yielded similar classification accuracy, but RVM performs better than SVM in terms of classification speed. This is because RVM generates smaller number of

relevant vectors compared to the number of support vectors generated by SVM for the same dataset size.

Goweder *et al.* [145] designed a hybridized anti-spam system based on MLP and GA. In the study, the authors considered emails containing English and Arabic words. They developed a classifier using extracted words with high TF-IDF value and trained the network using GA. They performed different experiments using three datasets obtained from SpamAssassin and TREC 2005 [156]. Two of the dataset contained 1000 emails. The third dataset contained 72 Arabic emails. The classifier was evaluated and it yielded an average classification accuracy of 94% (for spam emails) and 89% (for legitimate emails). Results revealed that Subject and Body section of emails is sufficient to design good spam detection system. Results also revealed that the following GA parameters are significant to the performance of NN classifier: population pool size, mutation method, crossover and mutation probabilities.

Manjusha and Kumar [146] designed a spam filtering system based on a combination of BN, ANN and GA, called BNNC. The authors used BN to classify email header information, ANN to process subject and body information. Furthermore, they used GA for feature selection and training. During training, they represented each email as chromosomes. Authors represented gene of each chromosome with different unique words extracted from header and body section of emails. These words are represented as 1 in the chromosome if they are in a blacklist and 0 otherwise. The authors calculated the fitness function for GA using conditional probability as explained in [146]. BNNC is composed of a network of Centred BN (CBN) and NNs. Each network is responsible for handling different parts of an email. Each CBN layer in the network represents the following parts in the header section of an email: From Address, From Name, attachment and CC. Also, each NN represent the subject and body section of an email. Each layer represents an external node. Each layer also has its Conditional Probabilistic Distributions, which defines update rules. Output for each CPD is binary. The binary output is sent to a priority based decision box, responsible for email classification. Each external node of CBN takes two values, either 0 or 1. Each node has internal nodes whose values are changed according to input from their respective external nodes and CPDs. The binary output from all layers in the network is sent to the priority decision box which classifies the email. The number of nodes in the network is equal to the number of vectors values extracted from data. Therefore, node corresponds to a unique word. BNNC also has one output layer, which displays either 0 or 1. The authors evaluated the technique on a dataset consisting of 2000 emails. During evaluation, they used 100 spam emails and 100 ham emails. The technique produced a TP rate of 1, FP rate of 1 and precision of 0.99.

Karthika and Visalakshi [117] used ACO in combination with SVM to provide solution to spam email detection problems. ACO was used for feature selection and SVM was used for email classification. The authors extracted email content from the dataset, tokenized the content and pre-processed them. They used ACO to select the best set of features. Finally, they passed the selected features to SVM for classification. They then compared the performance of the proposed hybridized technique (called ACO-SVM) to SVM, KNN and NB. ACO-SVM yielded best result. KNN, NB, SVM and ACO-SVM yielded a classification accuracy of 75.25%, 76.24%, 79.5% and 81.25% respectively.

Yin *et al.* [147] used Linear Discriminant Analysis and ACO to solve the problem posed by spam emails. LDA was used for feature selection and ACO was used to design classification rules based on selected features. The authors evaluated the technique on a dataset consisting of 2412 ham emails and 481 spam emails, and compared its performance to SVM and NB. LDA-ACO, SVM and NB yielded a precision of 96.83%, 94.76% and 89.48%.

Yang [148] used a combination of ACO, rough set and GA (called RCGF) to provide solution to spam email detection. The authors used ACO, RST and GA for feature selection. In the study, they proposed an algorithm (called AF algorithm) for feature selection. Classification was performed in three stages. In the first stage, AF algorithm combined with ACO and roughest theory was used to select a subset of features. Roughest was used to handle local search for ACO. The combined technique was used to produce a feature subset. In the second stage, GA was used to further optimize the selected features. In the final stage, the selected features were passed to SVM, KNN, ANN and NB for classification. The authors performed some experiments using two datasets obtained from PUI and Ling-Spam (combination of LING and SpamAssassin) respectively. The first dataset contained 481 spam messages and 618 legitimate messages. The LING dataset contained 481 spam messages and 2412 legitimate messages. SpamAssassin contained 1897 spam emails and 4150 legitimate emails. The authors compared the performance of ACO, PSO, GA and RCGF to each other. RCGF (combined with SVM) outperformed the other algorithms, producing the highest precision of 97.34%.

Zitar and Hamdan [149] used GA and NN to develop an anti-spam solution called Continuous Learning Approach ANN (CLA_ANN). In the study, the authors used NN for classification and GA to optimize the spam email classifier. Prior to classification, they extracted features from datasets, calculated their weight, and used selected features to train the network. Afterward, they tested the trained network. During testing, each email in the test dataset was tokenized, pre-processed and weighted. The weight of each email was then compared to a user-defined threshold. An email is considered to be spam if its weight is greater than the user-defined threshold, otherwise, it is considered to be ham. If the email is spam, all new tokens in the email will be added to a database, and used for future classification. Additionally, the authors used GA to periodically check whether an email status has changed from legitimate to spam. To achieve this, they firstly used GA to randomly generate a threshold value, which specifies the number of emails to be accommodated in an inbox. The threshold value is increased to accommodate new emails if the inbox is full. The authors trained the network on a dataset containing 1075 spam emails and 710 ham emails obtained from SpamAssassin. They also tested the network on a dataset containing 682 spam emails and 3435 ham emails obtained from SpamAssassin. The performance of the proposed technique was evaluated, and it yielded a classification accuracy of 98.86%.

Dhanaraj and Palaniswami [150] proposed an improved FFA-based spam email classification approach with an objective of improving computation time and feature space dimensionality of spam email filters. Computational time was improved by implementing the algorithm in a distributed environment, called Hadoop distributed environment. During classification, the authors used FFA for feature selection and NB for classification. They performed some experiments on two datasets, obtained from spambase and CSDMC2010 SPAM corpus. Spambase consists of 1794 spam emails and 2806 ham emails. CSDMC2010 SPAM corpus consists of 2949 ham emails and 1378 spam emails. 3601 emails were used for training and 920 emails were used for testing. During the experiments, the authors noted that the feature selection process and the classification process were distributed using Map-Reduce framework [157]. The proposed technique yielded an accuracy of about 80%. The authors compared the proposed technique to PSO and NN, and it performed better.

Zhang *et al.* [121] proposed a spam email classification technique based on BPSO and decision tree. In the study, the authors used Modified BPSO (MBPSO) for feature selection and decision tree for classification. Also, they used C4.5 algorithm to train the decision tree. During training, they performed cross validation and recorded the out-of-sample cost for each iteration. Finally, the out-of-sample cost for each iteration was summed, averaged and displayed to users. In the study, the authors introduced a cost matrix to assign different weights to errors from FP and FN. They evaluated the performance of the proposed technique on a dataset containing 6000 emails obtained from UCI ML repository. They also compared the performance of MBPSO to two existing feature selection techniques and, as reported, MBPSO performed better. MBPSO produced a classification accuracy of 94.27%.

Wu *et al.* [139] introduced a novel spam filtering framework based on PSO, SVM, F-Score and fuzzy logic. In the framework, SVM was used for classification, F-score and PSO was used for feature selection. The authors used F-score to calculate the importance of each feature, and used PSO along with some fuzzy controllers to further optimize the feature space. They tested the technique on three datasets obtained from PUI and Ling-Spam collection respectively. The first dataset (PUI) consisted of 481 spam emails and 618 legitimate emails. The second dataset (Ling-Spam) contained 481 spam emails and 2412 legitimate emails and the last dataset (SpamAssassin) consisted of 1897 spam emails and 4150 legitimate emails. The test produced a precision of 96.91% (on PUI), 97.83% (on Ling-Spam) and 94.21% (on SpamAssassin).

Wang *et al.* [151] developed a feature selection solution based on PSO and RST. In the solution, the authors used rough set to decompose a feature space into smaller subsets, and PSO to select optimal subset. Afterwards, the subset with fewer features and high classification accuracy was selected and used for training. During experiments, the authors divided the features space into 2N feature subsets, where N is number of features. They implemented the proposed technique along with four other feature selection algorithm, and compared their performance. They represented each feature as a binary number, where 0 indicates that the feature is selected and 1 indicates otherwise. Classification accuracy obtained from implementation ranged from 59.9% - 100%.

Salehi and Selamat [158] used SAIS in combination with PSO to determine solutions for spam email filtering. PSO was used for feature selection and SIAS was used for classification. The authors calculated TF of extracted features, normalized and saved them in a dataset. Afterwards, they selected 70% of the dataset for training and 30% for testing. Additionally, they divided the dataset, containing spam emails, into two: exemplar and training. Five percent of the dataset was used as exemplar, and 95 percent was used for training. The exemplar dataset was used to create an initial classifier system. Mutation was applied to the exemplar, and PSO was used to select 30% of the mutated classes. The process was repeated until all data in the training dataset, and classes in the exemplar, is considered. The authors performed experiments, and they produced a classification accuracy of 88.33%.

2.4.4 Survey discussion: Spam email detection techniques

This section presented some NI-based and ML-based techniques proposed by different authors seeking to improve the performance of spam email detection systems. Figure 2.1 reveals that GA and AIS are the most popular NI algorithms used for spam email detection. Also, results obtained in [56] revealed that GA is a good optimization technique suitable for optimal dimension reduction. GA is also a good algorithm suitable for optimizing SVM parameters, and improving SVM classification speed and accuracy [56]. GA can be used in combination with ML technique to design a robust spam email detection solution. For example, GA was used in [56] and [132] to optimize SVM parameters, and consequently improve SVM classification speed and accuracy. However, GA is time-consuming [121]; it requires more parameter tuning [127]; it cannot effectively search for a perfect solution [129], and it is not suitable for local optimization [129]. MA is an improved algorithm that has not been fully explored. It is a better algorithm (compared to GA) that can handle local optimization [129]. MA combines GA and a local search technique to comb for solutions [129]. Another effective feature selection technique is PSO. PSO is a better feature selection technique compared to GA [139]. It has fewer parameters compared to GA and it also does not have complex time-consuming operators like GA, such as crossover and mutation [139, 151]. In PSO, time is mainly consumed during fitness function evaluation [151]. Also, PSO is quicker in locating optimal solution compared to GA [151]. However, execution time of PSO is affected by data size and feature size [151]. As mentioned, AIS is one of the widely used NI algorithms for spam email detection. SAIS can be used in combination with other techniques to obtain better performance [149]. Authors in [149] and [158] used AIS in combination with NN and PSO respectively, and obtained promising results. Zitar and Hamdan [149] pointed out that the number of lymphocytes used in AIS-based techniques, affects system performance. Hence, to obtain good classification results, the number of lymphocytes used should be carefully chosen.

Figure 2.2 shows the Google Scholar report for the top six algorithms that has been used in literature, to handle spam email, between year 2010 and 2015. As shown in Figures 2.1 and 2.2, NB algorithm is the most widely used ML algorithm. Figure 2.2 reveals that NB has been widely and co consistently used since 2010. NB can be used in combination with other technique or heuristics to build an improved spam email detection system [144, 154]. However, Bayesian based techniques are vulnerable to Bayesian poisoning – a method used by spammers to bypass Bayesian based filters [124]. Also, performance of NB is affected by feature space of high dimensionality [118, 144] and change in class ratio (e.g. spam to ham ratio) [137]. In the survey, SVM-based techniques also yielded promising results. Figures 2.1 and 2.2 reveal the wide usage of SVM. Tseng and Ming-Syan [130] proposed an incremental update technique for SVM. The proposed technique is dynamic, and applicable to real world environment. However, the authors noted that the proposed technique is time-consuming. Classification speed and accuracy of SVM can be improved by using NI algorithms. Experiments performed by Yu and Xu [118] revealed that SVM is a better classification algorithm compared to NN and NB. It also revealed that performance of SVM is not affected by number of features; it is mainly affected by number of support vectors. RVM is another ML classification algorithm that yielded promising results in literature. However, as shown in Figure 2.1, RVM has not been fully explored. RVM is an effective classification technique, it is faster than SVM and its performance is not significantly affected by feature space dimensionality [118]. RVM also consumes more time for training compared to SVM [118]. Another algorithm, proposed in literature for feature selection, is RST. Wang *et al.* [151] noted that Rough set hill-climbing approach cannot adequately find perfectly reduced subset. They are efficient when applied on dataset with little noise and few features. Rough set stochastic techniques are more robust, but they are time-consuming [151]. Speed of Rough set based system can be improved by parallelizing computations of reducts [151].

NN is another popular ML technique that has been proposed in literature for spam email classification. However, NN is not a good stand-alone spam email detection technique [116, 145, 146, 149]. Also, training time for NN is high, and its accuracy is affected by dataset and feature size [145]. Speed of NN can be improved if it is combined with other optimization techniques, such as ACO. ACO is a good optimization and feature selection technique. El-Alfy [126] noted that increase in number of ants and other ACO parameters will increase computational time of ACO. Another promising feature selection technique that has not been fully explored in the literature is LDA. Yin *et al.* [147] are a few of the authors who have used this technique. The authors used LDA in combination with ACO to design an improved spam email filter. As obtained from Google scholar, Figure 2.1 reveals that EA and FFA has not been widely used to design spam email filters. The authors in [135] and [150] are some of the very few who worked on EA and FFA. Results obtained by Dudley *et al.* [135] revealed that heuristic-based systems that perform many tests consume time. Hence, classification speed of these systems can be improved by reducing the number of tests performed. Furthermore, results obtained by Dhanaraj and Palaniswami [150] revealed that FFA is a good speed optimization technique. It also revealed that computation time of spam filters can be reduced by using distributed systems.

Some of the proposed email filtering techniques are rule-based. Rule-based filtering systems, such as Ripper [159] and decision tree [160], can be easily bypassed by spammers, because they are dependent on specific terms (i.e. rules); hence, non-existence of the specified terms will lead to filtering failure [126, 127]. Furthermore, some of the proposed techniques are biased towards a particular email class. For example, the technique proposed by Xiao-li [153] is biased towards ham emails. Also, the technique proposed by Nosseir *et al.* [141] is biased towards classes with higher weights. Incoming email belonging to a category with higher weight is given higher priority than incoming emails with lower weight. A robust spam email classification technique should have negligible misclassification rate; it should be capable of effectively detecting both spam and ham emails. Most of the proposed techniques are keyword-based. However, behaviour-based feature proposed by Wu and Tsai [142] yielded the best result, in terms of classification accuracy. The authors noted that behaviour-based spam email filter is more effective than keyword-based filter, because the rate of change of keywords is higher compared to rate of change of spam behaviour. Furthermore, most of the proposed techniques did not consider attachments or images as part of features used for classification. Adding both features in spam email filters will undoubtedly improve classification accuracy of spam detection systems.

Some of the surveyed studies performed feature selection. Feature selection is essential: it saves computational resources and storage space [144]. Also, feature selection is better than feature extraction, because feature selection selects fewer features and consequently reduces computational complexity [121]. Feature selection also preserves useful rules [121]. Zhang *et al.* [121] noted that wrapper-based feature selection techniques are faster than filter feature selection techniques. Results obtained by the authors revealed that wrapper-based technique yield high classification accuracy. However, wrapper-based feature selection techniques are slow in execution and they lack generality [121]. The speed can be improved by using global optimization techniques and N-fold cross validation [121]. Some of the proposed techniques yielded poor results because they were trained on few data instances. Classifiers trained on large datasets would improve classification accuracy [120, 133, 143]. However, it may degrade classification speed. Classification speed can be improved by using distributed systems, feature selection and instance selection techniques. Many of the surveyed studies did not explore the use of distributed systems and instance selection. Studies in [150] and [138] is some of the very few studies that implemented distributed systems. Prilepok *et al.* [138] performed feature selection using GPU units – a distributed system.

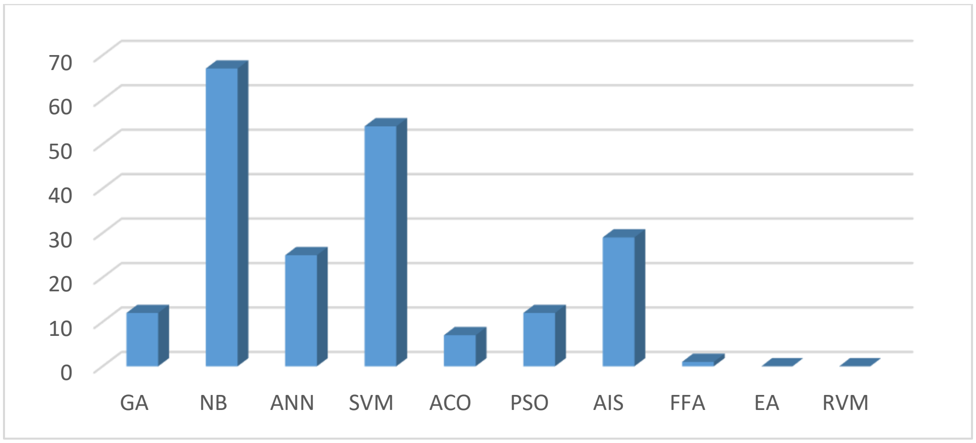

Figure 2.1: Spam email detection techniques between 2010 and 2015

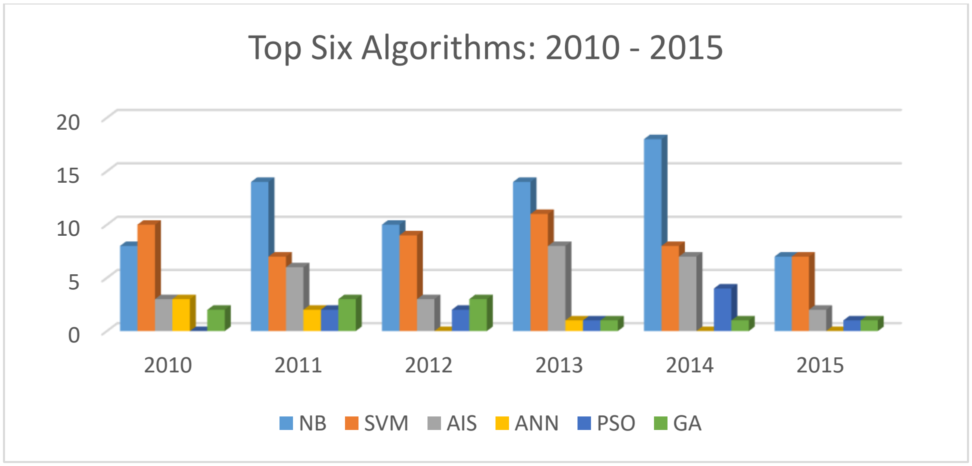

Figure 2.2: Top six spam email classifiers between 2010 and 2015

2.4.5 Limitation of spam email detection techniques

A sizable number of the reviewed spam email classification techniques did not achieve high classification accuracy, precision or recall. Also, some of the proposed techniques used traditional techniques (such as Term Frequency (TF) or IG) for feature selection. Sorayya and Seyed [122] pointed out that parameter optimization and feature selection are two effective techniques that have been proposed in literature to improve spam filters. Among all the reviewed techniques, to the best of the authors' knowledge, no proposed model used NI algorithms for both feature selection and parameter optimization. NI algorithms can be used to improve the performance of ML-based classifiers by reducing the feature space and parameter space dimensionality. Future work should focus on designing NI-based ML models with both feature selection and parameter selection techniques.

2.5 Credit card fraud

Credit card fraud can be defined as illegal use of credit card information for online purchase. Credit card transactions are done physically or virtually [161]. Physical transactions refer to transactions involving physical interaction with seller. Users are required to present a physical card at the point of purchase [161]. Virtual transactions refer to transactions performed over the internet or telephone [161]. They require users to provide certain card information (such as Card Verification Value or CVV number, password, security question, etc.) for online purchases [161]. The invention of credit cards has not only made online transactions seamless, easier, comfortable and convenient, it has also provided new fraud opportunities for criminals, and increased the rate of fraud [162, 163]. The effect of credit card fraud is alarming, and has affected the global economy in measurable ways. Millions of US dollars have been lost by many individuals and companies. In 2009, the total value of online order (for goods and services only) was approximately US$15 billion [163]. 84% of these orders were paid online [163]. In 2013, fraud was estimated to cost US retailers about $23 billion, and in 2014, the cost of fraud rose to approximately $32 billion [164]. Weak security of credit and debit card is one of the major causes of credit card fraud. In the UK, card-not-present fraud was estimated to cost £183.2 million in year 2011 [165]. Also, VISA processes are worth in transactions approximately US$3 trillion   every year, and for every $100, seven cents go to irregular transactions [166]. Every credit card user stands the risk of falling victim to card-not-present fraud and retailers bear the cost of irregular transactions [167].

Credit card fraud detection is a classification problem [165]. Credit card numbers are generated using Luhn algorithm [165]. The algorithm does not categorically protect users from online fraud; it essentially helps in authenticating data input from users [165]. Some small scale companies use manual authentication methods, including validation of phone numbers, physical address, secret question and answer [165]. However, these methods may not be feasible for large scale companies, and they are expensive and inefficient [165]. Most online merchants now use CVV2 as an additional security measure for approval of card-not-present transactions [165]. Although this additional security measure has reduced card-not-present fraud to a reasonable minimum, it does not prevent fraud that occurs due to lost or stolen card [165]. Address Verification Service can be used to combat card-not-present fraud. It is an electronic service that verifies transactions by using shipping address details of card owners [165]. This method reduces fraud; however, it can lead to loss in sales, because not all customers are willing to ship purchased items to their billing address [165]. MasterCard and VISA card has introduced a 3-D secured protocol for online banking: MasterCard Secure Code and Verified by VISA [165]. These protocols use a digital certificate to authenticate online merchants and password to authenticate customers [165].

Fraudsters mostly use internet to commit fraud, because their identity and location can be easily concealed [168]. Loss incurred from credit card fraud affects both customers and merchants. Although, merchants bear most of the loss, customers are made to pay higher interest rates and higher fees for membership [162]. Merchants also reduce their promos and incentives [162]. Fraud detection is absolutely essential in reducing losses incurred by financial institutions and individuals. The primary objective of fraud detection systems is to identify fraud promptly [163]. In a credit card transaction, four parties are typically involved: the card holder, merchant, financial institution and the VISA centre [169]. All these parties require security. Most of the existing fraud detection systems are rule-based system [170]. Rules are developed based on known patterns, hence these systems are only capable of detecting known fraudulent patterns; they are not capable of detecting unknown or emerging patterns. On average, it takes approximately 72 hours for a fraudulent transaction to be discovered [171]. Duman and Ozcelik [170] note that rule-based systems are only useful for counterfeit card fraud detection; they are not useful for lost/stolen card fraud detection. To address this issue, fraud detection system developers should take into cognizance fraudster behaviour and card user behaviour [170].

Some common types of fraud include credit card fraud, computer intrusion, money laundering [168]. This section presents a survey of some recent credit card fraud detection techniques proposed in the literature. Popular NI and ML credit card fraud detection techniques used in the literature include HMM, NN, SVM, AIS and GA. Other techniques include meta-learning, frequent pattern learning, ontology and decision support system. These techniques are used alone or hybridized with other techniques to construct robust classifiers. In some studies, NI algorithms were hybridized with ML algorithms, and in other studies, two or more ML algorithms are combined (called ensemble). Generally, hybridized techniques perform better than stand-alone techniques. Stand-alone ML-based credit card fraud detection techniques used in the literature include NN, HMM, Meta-learning, SVM, Frequent itemset mining, ontology, decision support system and Fisher Discriminant Analysis. Stand-alone NI-based credit card fraud detection techniques used in the literature include AIS and GA. Furthermore, hybridized techniques used in the literature include HMM and KNN, ANN and simulated annealing, decision tree and SVM, BN and NN, transaction aggregation and logistic regression. Few studies used Fisher Discriminant Analysis, simulated annealing, ontology and frequent itemset mining. Table 2.4 gives a summary of the surveyed techniques. This section presents a survey of some these techniques. It also outlines the contributions and limitations of the proposed techniques.

2.5.1 Machine learning based credit card fraud detection techniques

Some credit card fraud detection techniques have been proposed in the literature. However, most of the proposed techniques are based on supervised learning and few are based on semi-supervised learning. This section presents a survey of some existing ML-based credit card fraud detection techniques.

2.5.1.1 Hidden Markov Model

Khan *et al.* [172] proposed a technique based on HMM and K-clustering. In the study, the authors used HMM to model a sequence of credit card transactions and used K-clustering algorithm to divide the transactions into three clusters: high, low and medium. Afterwards, incoming transactions were compared to past ten transactions performed by card user and authorized if there was a match. Otherwise, the transaction will be terminated and IP address of the merchant to be defrauded will be traced using HMM. A notification will be sent to both the merchant system's administrator and mobile number of card owner. The authors noted that HMM was trained with Baum-Welch algorithm. They did not provide details about results obtained from the proposed solution.

Ashphak *et al.* [173] proposed a solution to credit card fraud detection system based on HMM. The system performs detection using spending patterns of cardholders. During classification, system request for card information of user and compares the information to information stored in a database. If there is a match, the system will request for PIN number of user. If the PIN is correct, and account balance is less than transaction amount, the system will ask user to provide answers to some secret questions. If the answers are correct, then an initial sequence of the users' 10 previous spending pattern will be extracted and passed to HMM for processing. Thereafter, HMM will calculate probability of acceptance for the new transaction. If the probability of acceptance revealed that there are no observed abnormalities, the transaction will be authorized. Else, if system observes some irregularities or if the number of transactions performed by the user is less than 10 transactions, then the user will be asked to provide answers to some security questions. If the answer provided is correct, the transaction will be performed in a secured mode; otherwise, the transaction will be terminated and referred back to the merchant's website. When a new transaction arrives, it is used to replace one of the old transactions in the sequence. The authors evaluated the performance of the proposed technique and it produced an accuracy of 92%.

Table 2.4: Summary of surveyed credit card fraud detection techniques

Technique Category	**Name of Technique**	**Reference(s)**
NI	AI	[165, 174, 175]
	GA	[176, 177]
ML	NN	[178, 179]
	HMM	[180-182]
	Meta-Learning	[183, 184]
	SVM	[185]
	Frequent Itemset Mining	[186]
	Ontology	[187]
	Decision Support System	[188]
	Modified Fisher Discriminant Analysis	[189]
Hybridized - NI and ML	HMM and K-Clustering	[172]
	ANN and Simulated Annealing	[190]
	Observation Probability and HMM	[173]
	BN and Neural Network	[162]
	Decision tree and SVM	[168]
	KNN + Decision tree + NB	[163]
	ANN and Logistic Regression	[191]
	Recency-Frequency-Monetary and time-dependent score	[179]
	Bagging and Ensemble	[192]
	Transaction Aggregation + Logistic Regression	[193]

Mhamane and Lobo [180] proposed a HMM-based fraud detection system. The system consists of 10 different modules. The first module allows users to interact with the system. In this module, users are allowed to login. The second module provides interaction between client and server. The third module allows a client to gain access to all items on the internet. The fourth module is responsible for authenticating transaction credentials entered by users. The module also generates a report if authentication is successful or not. The fifth module provides communication to servers via servlets. The sixth module is responsible for maintaining database of all account information of users. The seventh module maintains a database of past transactions already performed by users. The eighth module is responsible for performing classification of transaction. HMM is used to scan and classify transactions. The ninth module is for system administrators. It provides a Graphic User Interphase that allows admin users to login and view account information of clients. New clients can also be added. The tenth module allows admin users to see accounts that are blocked. Admin users can also reactivate blocked account and change credentials of users. The authors did not report on results obtained from study.

Bhusari and Patil [181] designed a fraud detection model based on the HMM and K-clustering. The authors used HMM to monitor spending patterns of users. When a user initiates a payment request, firstly, it will be submitted to merchant's system for processing. If the PIN entered by the user is correct, then the transaction amount will be compared to account balance of card holder. If the transaction amount is greater than the account balance, then the transaction will be denied and passed to a module responsible for fraud detection; otherwise, the transaction will be passed to the next stage for processing. Furthermore, with the aid of K-clustering algorithm, the authors divided the amount of previous transactions (stored in the dataset) into three price ranges (low, medium and high). HMM was used to check the last ten transactions (performed by the card holder) for abnormalities in spending patterns. HMM uses transition probabilistic calculation. If any abnormality is observed, the user will be asked some security questions. If wrong answers are provided, the transaction will be denied and HMM will raise an alarm to the issuing bank. The authors noted that if the number of transactions performed by the card holder is less than ten, then user will be asked some security questions. If provided answers are correct, user will be allowed to proceed with transaction. Some experiments were performed and it was reported that the proposed technique yielded an accuracy of 84% and a false alarm rate of 7%.

Mhamane and Lobo [182] proposed a HMM-based fraud detection technique. System architecture of the technique consists of the following component: legitimate user, fraudulent user, bank server and bank database. Bank database is used to store information about bank account holders. It is also used to store previous transactions of users. During training, the system extracts sequence of transaction details about users from dataset and builds a HMM-based classification model using the extracted details. The authors used trained models to classify incoming transactions. If there is a violation in the sequence of transactions, an One Time Password (OTP) will be sent to the mobile number of user. The authors evaluated the performance of technique and it yielded a classification accuracy of 72%.

2.5.1.2 Support vector machines based techniques

Sahin and Duman [168] performed a comparative study between SVM-based and decision tree based credit card fraud detection system. The authors used four kernels for SVM. During implementation, firstly, they divided datasets used into three groups. In the first, second and third group, the ratio of fraudulent transaction to legitimate transaction was 1:1, 1:4 and 1:9 respectively. In each group, 70% of the dataset was used for training and 30% was used for testing. The authors developed seven SVM-based and decision tree based models and tested each of them. Results from experiments revealed that decision tree based model outperformed the SVM model. The models produced classification accuracy between the range of 83.02% and 94.76%.

Lu and Ju [185] used PCA and Imbalanced Class Weight SVM (ICW-SVM) to develop a credit card fraud detection model. The authors used PCA for feature selection and used ICW-SVM for classification. Feature selection was achieved by calculating the principal components of all features and selecting features with the highest contribution rate. Selected features were then passed to ICW-SVM for classification. The authors noted that ICW-SVM handles data imbalance. Some experiments were performed and a classification accuracy of 91.28% was achieved. Furthermore, they compared the result to results of three other algorithms: BN, C-SVM and Decision tree (C5.0). ICW-SVM outperformed the three algorithms.

2.5.1.3 Meta-learning based techniques

Pun [163] designed a credit card fraud detection model. The author’s objective was to develop a classifier capable of filtering transactions for an existing Fraud detection system (called Falcon

Fraud Manager) used by major banks in Canada. The model consists of three base classifiers, constructed using k-NN, Decision tree and NB algorithm respectively. The author combined the output obtained from decision tree and K-Clustering and passed it to NB for classification. Classification is divided into four stages. In the first stage, authors trained the base classifier on 50% fraudulent transactions and 50% legitimate transactions. Afterwards, the author tested the trained base classifier on a validation dataset and generated some predictions. In the third stage, the author combined the generated predictions with validation dataset and used the combined dataset to construct a NB based meta-classifier. In the last stage, he tested the base classifier obtained in the first stage and combined the result with the test dataset. Furthermore, he used the combined dataset to re-train the meta-classifier. Results obtained from the re-trained meta-classifier are displayed as final output. The author performed some experiments to evaluate the performance of the designed meta-classifier and it yielded positive results. Additionally, the author compared the performance to performance of an existing bank's system, and it was reported that an improvement of 24% to 34% (resulting to a savings of $1.8 million to $2.6 million) was achieved.

Stolfo *et al.* [183] proposed a meta-learning based fraud detection system. The aim of study was to develop a distributed fraud detection system for financial institutions in a network. The distributed system will enable financial institutions share fraudulent models in a secured manner. The shared model will be combined by a meta-learner into a single robust meta-classifier. The technique consists of two main components. The first component (called local fraud detection agents) consists of four classifiers: ID3, CART, BAYES and RIPPER. The second component (a meta-learning system) combines outputs obtained from the individual classifiers to make a decision. In the study, the authors developed different classification models using ID3, CART, RIPPER and Bayes. The models were trained and tested using different datasets, and outputs from the best N classifiers were combined by a meta-learner to generate a meta-classifier. Bayes, RIPPER, CART and ID3 yielded a FP rate of 13%, 16%, 16% and 23% respectively.

Sen and Dash [184] investigated the performance of five meta learning algorithms in providing solution to credit card fraud detection. The algorithms include Classification and Regression tTree (CART), Adaboost, Bagging, Logitboost and Grading. Results revealed that the Bagging algorithm performed best (in terms of classification accuracy and misclassification rate) compared to the other four algorithms, while the grading algorithm performed worst. Bagging, Logitboost, Adaboost, CART and grading produced a classification accuracy of 87.7%, 85.5%, 84.7%, 83.4% and 53.6% respectively.

2.5.1.4 Frequent itemset mining

Seeja and Zareapoor [186] proposed a fraud detection technique capable of handling transactions in an imbalanced dataset. The authors also proposed a matching algorithm for classification of incoming transactions. During training, they extracted legal and fraudulent transaction pattern of all customers. Afterwards, they used the extracted patterns to construct a classification model. During testing, if an incoming pattern matches more with a legal pattern, then the transaction will be classified as legitimate, otherwise, it will be classified as illegal. The authors constructed two patterns for each customer - a fraud and legitimate pattern. Furthermore, they applied frequent itemset mining on transactions extracted from dataset. Frequent itemset mining evaluates transactions and returns different group of attributes. The group with the largest number of attributes is said to be the customer's legal pattern. During classification, the customer's details are extracted from database. Afterwards, legal and fraud transactions for each customer are separated. Frequent itemset mining algorithm is applied to the legal transactions of each customer, and the algorithm returns a set containing different group of attributes. Thereafter, the group with highest number of attributes are selected and stored in a database. Frequent itemset mining algorithm is applied to fraud transactions of each customer and the algorithm returns a set containing different group of attributes. Thereafter, the group with the highest number of attributes are selected and stored in a database. For an incoming transaction, a matching algorithm is used to scan the legal and fraud database. If an incoming pattern matches more with legal pattern, then the algorithm will classify the transaction as legitimate; otherwise, the algorithm will classify the transaction as illegal. The authors performed experiments and compared its performance to four other classifiers, SVM, RF, NB and KNN. Results revealed that the proposed technique yielded the best fraud detection rate.

2.5.1.5 Transaction aggregation

Jha *et al.* [193] proposed a credit card fraud detection technique based on transaction aggregation. The authors combined fraud and legitimate transactions of different time periods. Afterwards, they used aggregated transactions to create variables, which were in turn used to train a logistic regression model. They performed a series of experiments and a classification accuracy of 99% was achieved.

2.5.1.6 Ensemble based technique

Zareapoor and Shamsolmoali [192] proposed a credit card fraud detection model based on bagging ensemble classifier. The primary objective of study was to compare the performance of SVM, NB and KNN to the Bagging ensemble classifier based on Decision tree. The authors evaluated the performance of SVM, NB and KNN. They compared the result obtained to the Bagging ensemble classifier. Results revealed that the Bagging ensemble classifier yielded better fraud catching rate and false alarm rate. Result also revealed that the Bagging ensemble classifier is capable of handling data imbalance.

2.5.1.7 Ontology-based technique

Potamitis [187] in a Masters thesis, designed an ontology-based expert system for conceptualizing characteristics of existing fraud detection techniques and characteristics of fraud attacks. Specifically, the author designed the expert system to handle credit card fraud, bankruptcy fraud, credit card application fraud and 25 detection techniques. To achieve this, he first identified different fraud detection techniques from the literature. Furthermore, he analyzed the characteristics of the identified techniques and conceptualized the information into mathematical representations. Afterwards, he used the mathematical representations to build the ontology knowledge base system. He used the knowledge based system to design an expert system, andnoted that the expert system can assist software developers to choose techniques to implement for specific kind of fraud. He performed different tests on the expert system and it yielded excellent results.

2.5.1.8 Decision support system

Carminati *et al.* [188] developed an online fraud detection system called BANKSEALER. The system is based on a combination of semi-supervised and unsupervised technique. It builds models for different customer behaviour based on transactions stored in a database. During classification, BANKSEALER first weighs anomaly of each user transaction and then search for other users with comparable spending patterns. Lastly, the system measures the abnormality of current spending pattern of user relating to past spending pattern of user. BANKSEALER is currently deployed as a pilot project in a renowned Italian bank.

2.5.1.9 Modified fisher discriminant analysis based technique

Mahmoudi and Duman [189] proposed a novel credit card fraud detection technique based on Fisher Discriminant Function. The authors developed a modified version of the function. The modified version contains a weight function responsible for classifying transactions with higher financial cost implication. The authors developed five weight functions. The weight function compares available limit on a card to average limits on other cards, and assigns higher weights to cards with higher limit. They used Decision tree for feature selection. They evaluated the performance of the proposed technique and it yielded positive results.

2.5.2 Nature inspired based credit card detection techniques

NI algorithms has been used in combination with ML algorithms to provide solutions to credit card fraud detection. As shown in the survey, different NI and ML techniques has been used in literature, including: HMM, NN, SVM, AIS and GA. Among these techniques, survey reveal that GA are the most popularly used algorithms for credit card fraud detection. This section presents a survey of some existing NI-based credit card fraud detection techniques.

2.5.2.1 Artificial immune system based techniques

Wong *et al.* [165] proposed an AIS-based credit card fraud detection technique. The AIS system consists of six components: user interface, detector set, transaction processor, detector generator, database and automated testing machine. The user interface is responsible for accepting inputs (in form of transactions). It can also be used to check system status. The automated testing engine is responsible for sending transactions to system from a pool of transactions stored in a database. It is also used to save statistics-related data about the system's performance in the database. A detector generator is used to produce mature detectors (using negative selection) and memory detectors. It is also used for evolution of memory detector. A transaction processor is used to process and classify transactions. During implementation, the authors extracted data from datasets and mapped them into a bit pattern using a matching algorithm. Afterwards, they used the matching algorithm to classify transactions. The AIS system consist of the following: a representation and matching algorithm, negative selection algorithm, vaccination algorithm and an algorithm for memory cell evolution. Representation and matching algorithms were used to classify transactions; vaccination algorithm was used to reinforce the system's learning ability to adapt to evolving patterns. Negative selection algorithm was used to generate mature detectors – which were used to classify transactions. Matching algorithm consists of rules created for different transactions extracted from the database. During classification, output for each rule was combined and mutated. The authors designed the mutation function using GA. Furthermore, the mutated rules were compared to incoming transactions. If there was no match, the rules were destroyed and new set of rules were created. Otherwise, the rules were passed through negative selection process to ensure that they are self-tolerant. If they are self-tolerant, then they are kept; otherwise, they are destroyed and a new set of rules is generated. The authors tested the performance of the proposed technique and it produced a classification accuracy of 71.3%.

Soltani *et al.* [174] proposed an AIS-based fraud detection algorithm. Algorithm used clonal selection to create detectors. In the study, the authors generated fraud and normal detectors for all classes and used KNN algorithm for classification. Furthermore, they calculated Euclidean distance for all records in the database and selected records with the lowest distance as the k neighbors. They performed some experiments and a promising result was achieved.

Soltani and Akbari [175] proposed an improved credit card fraud detection model based on AIS. During memory cell generation, distance between each training records and their corresponding ARB (Artificial Recognition Ball) is calculated. Afterwards, records with low distance are selected for mutation. If the selected record belongs to the same class, it is selected for mutation. Otherwise, records with large distances in the same class are selected. At the end of memory cell generation, each cell is ranked based on its distance between each record it matches. If a memory cells performs wrong classification, it will be rated based on its distance between the wrongly classified records. Rating is performed using KNN algorithm. Authors explained that rank will be positive if memory cell and matched records belong to the same class; otherwise, rank will be negative. The authors tested the model and it yielded a detection rate and FP rate of 0.518 and 0.017 respectively.

2.5.2.2 Genetic algorithm based techniques

Patel *et al.* [176] proposed a GA-based credit card fraud detection system with the aim of reducing the amount of credit accessible to fraudsters. The authors defined an objective function with variable misclassification cost. The objective function aims at reducing the number of transactions with high classification cost. During classification, they extracted credit card transactions from dataset and stored them in a database. Afterwards, they calculated critical values for each transaction. They also extracted the following from each transaction: frequency count for credit card usage, credit card overdraft, location where the credit card was used, balance on account linked to credit card, average spending pattern of the credit card owner. Furthermore, the authors used the GA to generate new critical values. Finally, the new critical values were then used for classification. The authors did not report results obtained from study.

Duman and Ozcelik [170] introduced a hybridized credit card fraud detection system capable of handling misclassification cost. GA and Scatter Search algorithms were combined and used to build a robust credit card fraud detection algorithm called GASS. The authors worked with 43 parameters and a population size of 50. Forty seven of the 50 solutions were determined by generating 47 random numbers for 43 parameters. The remaining 3 solutions were solutions for generating maximum number of alerts (MAX), minimum number of alerts (MIN) and solution used for production (PRD). In the reproduction stage, the authors combined parameter values of two parent solutions to obtain a child solution. They noted that the reproduction process is different from the reproduction process of GA but similar to SSA. They also noted that the classification steps of GASS is similar to standard GA, but with some element of the SS algorithm. They carried out several experiments to evaluate the performance of the proposed technique, and results showed that GASS algorithm improved the performance of an existing fraud detection system by 200%.

RamaKalyani and UmaDevi [177] proposed a GA-based credit card fraud detection technique with varied misclassification cost. The objective of study was to limit the total amount of credit accessible by fraudsters. During classification, the authors extracted the following information from dataset: frequency of credit card usage, the location of usage the credit card overdraft, the available balance in the credit card and the average amount spent per day. Afterwards, authors used GA to generate critical values and also generate fraud transactions. Thereafter, new transactions are compared to the generated critical values and classified accordingly. They repeated the process until a user-defined threshold was reached.They tested the performance of technique and it yielded positive results.

2.5.2.3 Artificial neural network based techniques

Khan *et al.* [194] used simulated annealing and NN to develop a credit card fraud detection technique. The authors used simulated annealing to control parameters in the NN. They generated a random weight for all connections in the NN, and normalized them using TANH activation function. Afterwards, the authors created a weight matrix and randomized the matrix using simulated annealing. Furthermore, they generated new weights from output obtained from previous circle. They compared the weights to previous weights and updated them if there was an improvement. They also reduced the temperature after each iteration and compared it to a user-defined temperature. If the temperature is lower, the process will be repeated again. The authors evaluated the performance of technique and it yielded a classification accuracy of 89.6%.

Maes *et al.* [162] performed a comparative study between ANN and BN for credit card fraud detection. In the study, the authors extracted features from dataset, pre-processed and normalized them. Afterwards, they used the features to construct a BN and ANN-based models. They used STAGE algorithm to select the optimal configuration for ANN, and conducted different experiments. Results revealed that BN outperformed ANN in both classification speed and accuracy. However, the authors pointed out that fraud detection process of ANNs is faster.

Modi *et al.* [178] constructed a NN rule-based fraud detection system capable of providing solution to credit card fraud. The authors used a single layer feed forward NN algorithm. In the study, they divided fraudulent transactions into four groups, namely, low, high, risky and high risk. Transactions are classified based on defined rules. If a processed transaction is fraudulent, it will be assigned to any of the four groups. The authors evaluated the performance of the algorithm. However, much detail about the results was not reported.

Kumar and Vasanth [191] developed a credit card fraud detection model based on ANN and logistic regression. The authors considered a classification problem with variable misclassification cost. Also, they used GA to optimize classifier parameters. During classification, they identified spending pattern of cardholder, computed some set of probability and constructed some sequence. Finally, they used the sequence to construct a NN-based and logistic regression based model.

Van *et al.* [179] proposed a novel credit card fraud detection technique called APATE. The technique combined two features. The first feature is based on characteristics of incoming transactions and spending history of customers. The authors used Recency-Frequency-Monetary

(RCF) fundamentals to derive this feature. The second feature is a time-dependent score based on network used by card holders and merchants. Incoming transactions are classified based on the following features: average number of past transactions over a time period, average time interval between incoming and previous transaction, and the value of the transaction. Incoming transactions are also classified based on a score indicating merchants frequently linked to fraud. Incoming transactions are also classified based on credit card holders with stolen cards or card holders that seldom perform transactions. The authors combined all the features and designed 78 variables which were used to construct three classification models based on logistic regression, RF and NN. They performed some experiments and reported that an AUC score higher than 0.98 was obtained. RF, NN and logistic regression yielded a classification accuracy of 98.7%, 93.84% and 95.92% respectively.

2.5.3 Survey discussion: Credit card fraud detection

The surveyed techniques reveal that various ML and NI algorithms have been used to handle credit card fraud detection. As shown in Figure 2.3, google scholar reveals that HMM, NN, SVM, AIS and GA are the most popularly used algorithms in the domain of credit card fraud detection. Furthermore, among these algorithms, as shown in Figures 2.3 and 2.4, google scholar reveals that HMM and NN have gained more attention and they have been used consistently for the past four years. These algorithms are used alone or in combination with other techniques, such as meta-learning or ensemble techniques. HMM is simple to implement; it removes classification complexity and it can be used to produce simple classification models [173, 181]. The training time of ANN takes several hours [162], sometimes days [194]. NN-based algorithms require parameter tuning algorithm (such as GA) and an effective algorithm for good network configuration [194]. Furthermore, some authors used Meta-classifiers, which yielded good results [163, 183]; however their classification speed is slow because they involve combination of several classifier. Moreover, Fisher Discriminant Analysis is one technique that has not been fully explored in the domain of credit card fraud detection. Technique proposed by Mahmoudi and Duman [189] is one of the few techniques that used Fisher Discriminant Analysis. The technique was designed to maximize high value transactions and FNs. Experiments performed in the study yielded good results, implying that Fisher Discriminant Analysis is a promising algorithm to explore. Another area that has not been explored is ontology. Potamitis [187] is one of the few that designed an ontology-based technique. Potamitis [187] introduced an ontology-based expert system for conceptualizing characteristics of existing fraud detection techniques. However, the technique is static; it requires regular update of knowledge base.

As mentioned, NI techniques have been used to provide solution to credit card fraud detection problems. NI techniques are capable of improving classification speed and accuracy of ML algorithms. Authors in [165], [174] and [175] proposed AIS-based credit fraud detection techniques. AIS-based systems aim to model a representation of detector and antigen relationship [165]. Afterwards, a matching algorithm is required to determine the strength of affinity between the antigen and detector. However, unlike AIS, matching algorithms are not capable of detecting non-self-organisms [165]. AIS is commonly used to model negative selection [165]. Wong *et al.* [165] noted that AIS-based techniques are not dynamic. Authors handled this limitation by designing a dynamic AIS-based system that models fraudulent patterns in e-commerce systems. Also, Soltani and Akbari [175] introduced an improved credit card fraud detection model with a modified method for performing negative selection. However, the memory generation phase and calculation of affinity are time-consuming. Additionally, Soltani *et al.* [174] proposed a novel credit card fraud detection model capable of handling misuse and anomaly detection. However, FP rate of the model is too high and generating detectors for all transactions can affect the classification speed.

As mentioned, the GA is one of the popular NI algorithms that have been used to handle credit card fraud. Authors in [176, 177] used GA to improve credit card fraud detection. Rinky and Dheeraj [176] used GA to generate nodes and hidden layer for NN. Duman and Ozcelik [170] used GA in combination with SSA to design a fraud detection technique with new classification cost function. Authors in [177] and [191] used GA for parameter tuning. However, experiments performed by Duman and Ozcelik [170] revealed that GA's convergence rate is slow, especially when applied to large datasets. Furthermore, authors in [170, 176] and [177] proposed techniques for handling misclassification cost. Duman and Ozcelik [170] noted that data mining algorithms cannot effectively handle classifications with misclassification costs. Although high value transactions has more impact, low value transactions should not be underestimated. This is because a system can be compromised if multiple low value transactions are performed.

Moreover, many ML techniques have been used to handle credit card fraud. Authors in [172], [173], [180], [181] and [182] used HMM. Khan *et al.* [172] used HMM in combination with K-clustering. Authors used HMM to model sequence of credit card transactions. Authors trained HMM with Baum-Welch algorithm. Additionally, Mhamane and Lobo [182] proposed a ML-based technique for handling internet banking transactions. Authors used OTP as an additional security feature. ANN and BNN are two other ML techniques that have been explored in the literature. As mentioned above, NN-based techniques are generally slow. Maes *et al.* [162] performed a comparative study between BNN-based and ANN-based credit card fraud detection techniques and results revealed that BNN has higher classification speed compared to ANN. Authors suggested that ANN can be improved by removing connections and perceptron that are not used in training and performing weight updates. Radial basis networks and SVMs are good algorithms that can be used for weight updates [162]. ANN also requires effective algorithms for performing parameter selection [162, 194]. SVM is another ML algorithm that has been used to solve credit card fraud detection. The performance of SVM improves as the number of data size increases [168]. Lu and Ju [185] designed a SVM-based technique capable of handling classification that requires assigning variable weights to different classes. The authors noted that adjusting class weights can improve the classification speed and accuracy of a classifier.

Decision trees is one of the ML algorithms that has not been fully explored in the domain of credit card fraud. One of the few authors that have used decision tree is Sahin and Duman [168]. Authors performed a comparative study between SVM-based and Decision trees based credit card fraud detection systems, and results revealed that Decision tree outperformed SVM. Meta-learning technique is another approach that has been used to tackle credit card fraud. Pun *et al.* [163] proposed a technique based on meta-classifier model consisting of three classifiers, KNN, Decision tree and Bayesian algorithm. The authors noted that the technique was deployed in series with an existing bank's system and it yielded an improvement of between 28% and 34% performance. Stolfo *et al.* [183] also proposed a meta-learning technique. The technique consists of two main component. The first component (called local fraud detection agents) consists of four classifiers: ID3, CART, BAYES and RIPPER. The second component is a meta-learning system that combines the outputs obtained from the individual classifiers to make a decision. Results obtained from many of the proposed meta-classifier models are good; however, as mentioned above, classification speed of meta-classifiers is slow because it involves combination of outputs from two or more classifiers. Also, experiments performed by Stolfo *et al.* [183] revealed that TP and FP rate of meta-classifiers increases as labelled fraud data samples increases. Experiments revealed that a balanced dataset will yield an improved classification accuracy [183]. Additionally, experiments revealed that the best meta-classifier is BAYES [183].

Most of the existing studies focused on the classification of customer spending profile analysis and derived attributes [186]. However, few studies focused on classification of anonymous dataset. Two of the few authors who worked on this are Seeja and Zareapoor [186]. The authors proposed a credit card detection technique capable of handling transactions in an imbalanced and anonymous dataset. The technique has a good and balanced classification rate; however, fraudulent and legal patterns formed for customers and stored in a database requires regular updates. Furthermore, the authors noted that proposed technique cannot detect transactions with similar fraud and legal patterns. Another unique technique proposed in literature is Jha *et al.* [193]. The authors proposed a technique based on aggregation of transactions. In the study, they combined legal and fraudulent transactions and used the combined dataset to construct a classifier. They explained that both patterns were combined to capture the difference between buying behaviour of customers. They also noted that fraud detection involving large dataset requires dataset grouping and creating new attributes. In another work, Van Vlasselaer *et al.* [179] introduced a technique that combines two group of features. The first group of features (called intrinsic features) was obtained from incoming transactions and spending history of customers. The authors used Recency-Frequency-Monetary (RFM) fundamentals to obtain this group of features. The second group of features was obtained by calculating a time-dependent score based on the network of credit card holders and credit card merchants. The authors used NN, logistic regression and RF to test model and RF yielded the best result.

To summarise, most of the proposed techniques yielded promising results. However, most of the datasets used are very imbalanced. Most datasets contained higher percentage of legal transactions compared to fraudulent transactions. Furthermore, most of the proposed techniques were not tested on real-world dataset; they were tested on artificially generated dataset. This is because most financial institutions do not release datasets due to confidentiality agreements they sign with their customers. Additionally, classification speed and accuracy of most of the techniques were low. Most authors did not explore the use of NI techniques.

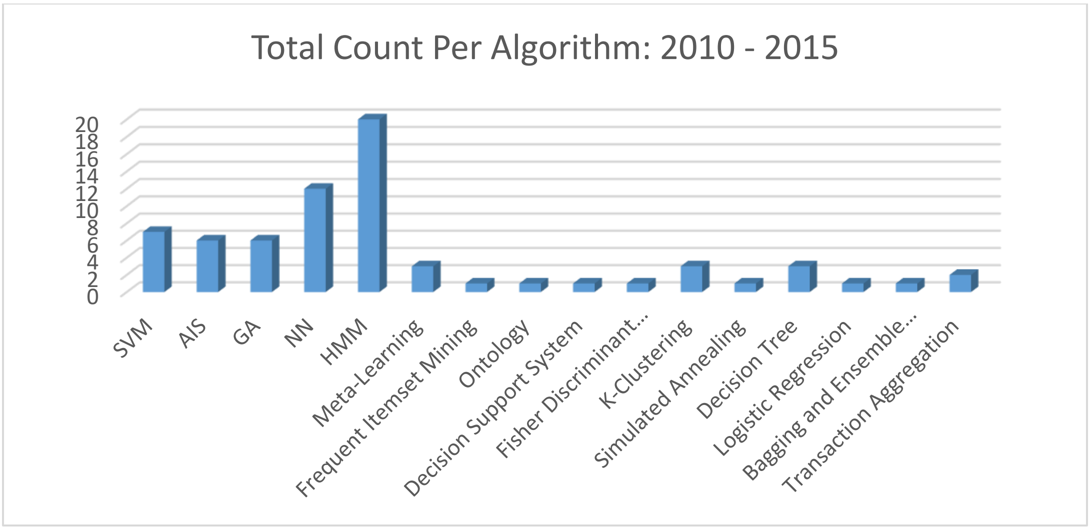

Figure 2.3: Existing credit card fraud techniques between years 2010 – 2015

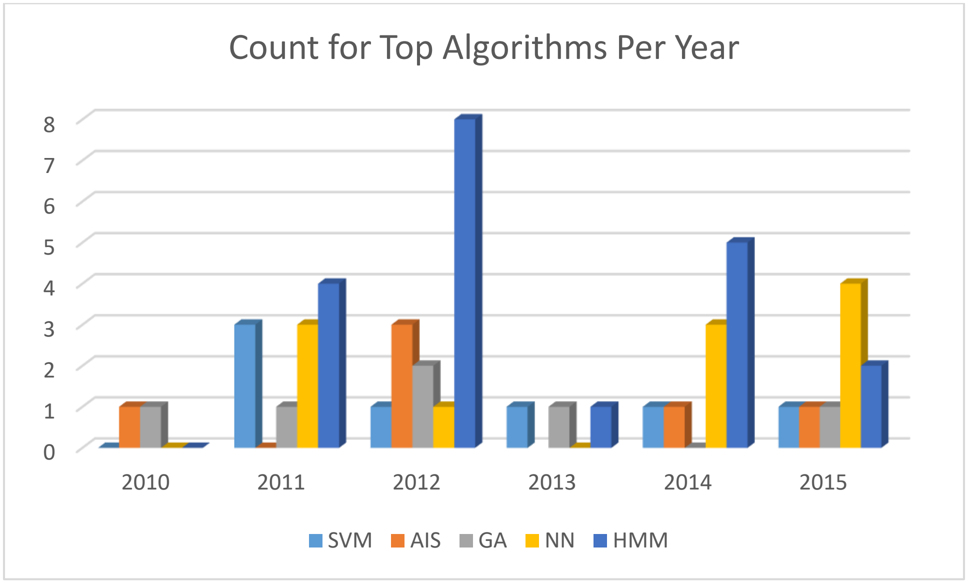

Figure 2.4: Number of proposed techniques for top six algorithms per year

2.5.4 Limitations of credit card fraud detection

Credit card detection is a fascinating domain. However, much work has not been done. The few authors that have worked in this domain provided little or no details on dataset used, features used and results obtained in their studies, making it very difficult to develop new techniques. Furthermore, many authors made use of imbalanced dataset [168, 170], and many of the credit card detection techniques surveyed in this paper used ML algorithms [162]. Many of them yielded low classification accuracy, FP rates and False Negative rates [165, 173, 190]. This is likely because the techniques were not combined with good and effective feature selection and parameter optimization technique. NI algorithms can be used to improve the classification speed and accuracy of credit card fraud detection system. Future work should focus on designing classification models capable of handling variables with different misclassification cost, and should consider focusing on constructing accurate classification models based on NI-techniques. This will likely increase the performance of credit card detection solutions.

2.6 General recommendations

As shown in the literature survey, many e-fraud detection and SVM speed optimization techniques has been proposed in literature. Some studies used NI algorithms, including AIS, PSO, GA, SSA, BA and FFA. Others studies used ML algorithms such as ACO, KNN, Clustering, BN, Decision tree, SVM, ANN and RF. Additionally, some studies combined different algorithms. Some studies utilized static approach and others utilized dynamic approach. Static approaches, such as blacklist, whitelist and rule-based systems, should not be used as standalone techniques, because of their inefficiency in handling emerging attacks. Static approaches can be used in combination with other techniques. Furthermore, some techniques produced good results, but they could perform better if some factors (such as dataset size, feature size and parameters), were properly considered. Based on the literature survey, the following are some helpful recommendations that can be considered when designing SVM optimization and e-fraud detection techniques.

1. Some of the proposed techniques yielded poor results because they were trained on few data instances. Dataset size used for training and testing in some studies is insufficient, for example, [195], [110], [195] and [78]. Email servers in real world scenarios store large volume of emails, hence email classifiers should be trained on sufficient number of data instances. Classifiers trained on large dataset would improve classification accuracy [120, 133, 143].
2. Feature size used in some studies is large. Generally, the performance of a classifier is determined by the quality (not quantity) of features used in training the classifier. Hence, instead of using large number of features for training, it is highly recommended that a reduced set of features is used. Feature selection techniques can be used to select relevant

features from a large feature set, which will consequently improve the overall performance of classifiers.

3. Most authors did not explore instance selection. Instance selection techniques are designed to reduce number of training instances by removing redundant instances from a training dataset. Instance selection is particularly useful for instance-based classifiers, where classification of one instance involves the use of an entire training set [6]. NI-based algorithms can be used to design effective instance selection techniques.
4. Speed optimization should be one of the main focus areas when designing email classifiers. Toolan and Carthy [106] designed an ensemble method for phishing detection (called R-boost). The authors noted that ensemble methods are not effective phishing detection techniques compared to some classifiers, such as SVM. Although R-boost outperformed almost all other techniques in literature, it is computationally expensive. This is because R-Boost requires at least four classifiers in the ensemble for the classification of just one sample.
5. Classifiers that require input from external sources should be avoided. Slow network communication from external sources can significantly affect classification speed. Also, inaccurate result from external sources can affect the accuracy of the overall classifier.
6. GA is not a fast algorithm for email classification. GA is time consuming [121]; it requires more parameter tuning [127]; it cannot effectively search for a perfect solution [129] and it is not a good candidate for local optimization [129]. Memetic algorithm is an improved and better algorithm (compared to GA) that can handle local optimization [129]. However, local search of memetic algorithm is affected by the random order used by scores in genome when performing optimization [129].
7. PSO is a better feature selection technique compared to GA [139]. It has fewer parameters compared to GA and it also does not have complex time-consuming operators like GA, such as crossover and mutation [139, 151]. PSO is quicker in locating optimal solution compared to GA [151]. However, data size and feature size affect the execution time of PSO [151]. PSO is quicker in locating optimal solution compared to GA [151].
8. Email classifiers should not be too complex. Algorithms used for designing email classifiers should be carefully chosen. For example, classifiers like Bayesian Classifier is

not mostly suitable for spam email classification [196]. Bayesian based techniques are vulnerable to Bayesian poisoning – a method used by spammers to bypass Bayesian based filters [124].

9. Performance of NB is affected by large feature space [118, 144] and change in class ratio (e.g. spam to ham ratio) [137]. Hence, number of features used to train NB should be taken into proper consideration. A feature selection technique is highly recommended.
10. RVM is a good classification technique; it is faster than SVM and feature space dimensionality does not significantly affect its performance [118]. However, RVM consumes more time during training compared to SVM [118].
11. NN is not a good stand-alone spam email detection technique [116, 145, 146, 149]. Furthermore, NN requires more training time, and its accuracy is affected by number of instances and input features [145]. Hence for better performance, NN can be used in combination with NI optimization techniques.
12. Rule-based systems are not capable of effectively handling emerging attacks. They require regular updates and can be easily bypassed by sophisticated attacks, because they are dependent on specific terms (i.e. rules). Hence, rule-based systems should be used as a supplement to dynamic techniques, such as NI-based and ML-based techniques.
13. Some of the proposed techniques are biased towards a particular email class. A robust email classification technique should be capable of effectively handling both classes.
14. Most of the existing spam email techniques are keyword-based. Wu and Tsai [142] noted that rate of keyword change is high, hence key-word based filters can be easily bypassed if not updated regularly. Behaviour-based features may be a better alternative to keyword-based features. Rate of change of behaviour-based features is lower compared to keyword-based features [142].
15. Most of the surveyed studies did not explore distributed systems. Computational speed of email classification can be greatly improved by implementing email filtering systems in distributed environments. In a distributed environment, different tasks are shared among different system in the environment, and the implementation of each task executes in parallel (or runs simultaneously). This approach is highly recommended, especially for huge data processing.

16. Static classification techniques, such as blacklist and whitelist, should not be used as stand-alone techniques. Blacklist and whitelist require regular update [75], and they are known to yield high FP rates. Moreover, both techniques cannot effectively detect zero hour phishing attacks [76], and they require more human resources [77].

17. Classification accuracy of spam email filtering systems depends on the number of overlapping words in different classes [143]. If two classes has too many overlapping words, the accuracy will be negatively affected [143]. Hence, prior to training, overlapping words should be reduced to the barest minimum.
18. Some of the proposed technique did not perform cross validation. Cross validation is very important, because it will correct the statistical dependency of all individual instances in the dataset [75], and it will also lead to a good and accurate estimate of evaluation.
19. Tradeoff between speed and accuracy should be taken into proper consideration when designing an email classifier. A good email classifier should be capable of efficiently classifying emails without significant degradation in classification accuracy.
20. Credit card fraud detection systems usually process millions of transactions. Hence, to improve the classification performance of fraud detection systems, there is a need for robust data dimension reduction technique and feature selection technique. NI algorithms are good data reduction techniques.
21. System developers can consider using HMM. It is simple to implement; it removes classification complexity and it can be used to produce simple classification models [173, 181].
22. Misclassification cost should be handled with care. Although high value transactions have more impact, low value transactions should not be underestimated. This is because a system can be compromised if multiple low value transactions are done. Researchers should focus on designing algorithms that can handle classification tasks with variable misclassification cost.

2.7 Chapter summary

This section present a comprehensive literature survey of existing e-fraud detection techniques and also provide detailed information on the current-state-of-the-art on e-fraud detection. The techniques reviewed in this section are divided into three categories: credit card fraud detection techniques, spam email detection techniques and phishing email detection techniques. As shown in the review, many e-fraud detection techniques has been proposed in literature, however, ML-based techniques produced the best result. ML algorithms are very good classification tools, nevertheless, their performance is significantly affected by large increase in dataset size. Section 2.1 provide a review of some existing speed optimization techniques that has been proposed in literature. As shown in the review, three major speed optimization approaches has been adopted in literature, namely: feature selection, parameter optimization and instance selection. Among the three approaches, instance selection methods produced one of the best results [6]. Moreover, as shown in Table 2.1, most of the existing instance selection techniques focused on clustering algorithm, KNN and EA [1, 53, 54]. Very few studies explored nature-inspired algorithms, and nature-inspired algorithms has the ability to efficiently find optimal solution to optimization problems. Therefore, this research propose NI ML-based models for e-fraud detection and classification problems. The design of the proposed techniques are discussed in Chapter 3.

Chapter 3
Proposed Techniques

This thesis proposes seven filter-based and five wrapper-based intelligent instance selection techniques for improving SVM training speed and predictive accuracy. This thesis also proposes a novel fitness function for instance selection. The filter-based techniques are designed for applications that require fast processing of large datasets, and the wrapper-based techniques are designed for applications that are very sensitive to a slight drop in classification accuracy. The main difference between the filter-based and wrapper-based techniques is in their method of selection. The filter-based techniques utilizes the proposed fitness function for instance selection, while the wrapper-based techniques utilizes SVM algorithm for instance selection. The primary objective of the filter-based techniques is to improve the training speed and consequently the computational complexity of SVM. The primary objective of the wrapper-based techniques is to improve the predictive accuracy and training speed of SVM. The filter-based techniques consist of seven instance selection techniques. The first two techniques are boundary detection algorithms, and they perform two major actions: boundary detection and instance selection. The two techniques are inspired by edge detection in image processing and edge selection in ACO. The remaining five filter-based techniques are inspired by the following NI algorithms: FPA, FFA, CSA, SSA and BA. The wrapper-based techniques consist of five instance selection techniques, inspired by FPA, FFA, CSA, SSA and BA. A brief introduction to SVM is presented next.

3.1 Support Vector Machines preliminaries

SVMs [197] are well-known classification and regression algorithms with a strong theoretical background. They can be used to handle both linear and non-linear classification problems. SVM performs linear classification using linear hyperplanes, and performs non-linear classifications using kernel functions. This section provides a brief introduction to SVM.

3.1.1 Linear support vector machine

As shown in Figure 3.1, Linear SVM (or hard margin SVM) can be used to classify instances that are linearly separable. Also, linear SVM can be used to classify instances that are not separable (soft margin SVM). Both cases are presented next.

3.1.1.1 Separable case

Given a training dataset, T = $[(x_i, y_i), \ldots \ldots, (x_n, y_n)]$, where x_i represent the vector values for each feature in the dataset, y_i represent the class labels. $x \in R^n, y \in [\pm 1]$. SVM aims to find a hyperplane with the widest possible margin. That is, the hyperplane that separates the positive class from the negative class. A hyperplane margin is computed by computing the distance between the closest positive class to the hyperplane and the closet negative class to the hyperplane. Hyperplanes with large distance (i.e. wide margin) are more resistant to noise compared to hyperplanes with smaller margins [29]. It is assumed that all data satisfy the following constraints:

$$\omega . x_i + b \geq +1 \quad y_i = +1 \tag{3.1}$$

$$\omega . x_i + b \leq -1 \quad y_i = -1 \tag{3.2}$$

where ω is the vector values in the higher dimensional plane and b is the bias (i.e. the offset value of the hyperplane). The two constraints (equations (3.1) and (3.2)) can be combined to yield the following [198]:

$$y_i(\omega . x_i + b) \geq 1 \ \forall_i \tag{3.3}$$

Furthermore, the margin m for each hyperplane can be computed using equation (3.4).

$$m = \frac{|1-b|}{\|\omega\|} - \frac{|-1-b|}{\|\omega\|} = \frac{2}{\|\omega\|} \tag{3.4}$$

The best margin can be computed by finding a solution to the following primal optimization problem [198]:

$$min_{\omega \in \mathcal{H}} \, \tau(\omega) = \frac{1}{2} \ \|\omega\|^2 \tag{3.5}$$

Subject to: $y_i(\omega . x_i + b) \geq 1 \ \forall_i$

For easier computation, the optimization problem can be re-formulated using the Lagrangian. The new optimization problem is given in equation (3.6).

$$min_{\omega,b} L(\omega, b, \alpha) \equiv \frac{1}{2} \ \|\omega\|^2 - \sum_{i=1}^{L} \alpha_i y_i (x_i \omega + b) + \sum_{i=1}^{l} \alpha_i \tag{3.6}$$

where α is the Lagrangian multiplier. Generally, some compulsory conditions must be satisfied for a non-linear programming solution to be optimal. This conditions are referred to as Karush-Kuhn-Tucker (KKT) conditions [199].

3.1.1.2 Non-separable Case

As aforementioned, some classification tasks cannot be effectively handled by linear classifiers, especially classification involving non-separable datasets. Soft margin SVM [200] was introduced to handle this type of classification. Soft margin SVM permits some mislabeled instances and then pays the cost for each mislabeled instance by adding slack variables, ξ_{i} to the re-formulated optimization problem defined in equation (3.6). This leads to the following equation [198]:

$$\forall_i \begin{cases} \omega \,.\, x_i + b \;\geq 1 - \xi_i & y_i = +1 \\ \omega \,.\, x_i + b \;\leq -1 - \xi_i & y_i = -1 \\ \xi_i \;\geq 0 \end{cases} \tag{3.7}$$

The addition of slack variables (as shown in equation (3.7)) causes some instances to fall within the decision boundary. Although, this makes SVM more robust to outliers, large slack variables can affect the optimality of a solution. Therefore, the original objective function (defined in equation (3.5)) can be modified to cater for slack variables. This leads to the following optimization problem [198]:

$$min_{\omega \,\epsilon\, \mathcal{H}, \xi \,\epsilon\, \mathbb{R}^m}\, \tau(\omega, \xi) = \frac{1}{2}\, \|\omega\|^2 + C \textstyle\sum_{i=1}^{m} \xi_i \tag{3.8}$$

Subject to: $\forall_i \begin{cases} \omega \,.\, x_i + b \;\geq 1 - \xi_i & y_i = +1 \\ \omega \,.\, x_i + b \;\leq -1 - \xi_i & y_i = -1 \\ \xi_i \;\geq 0 \end{cases}$

where C is a user-defined cost parameter that states the penalty that should be assigned to instances that are misclassified. The parameter C must be a positive value. Similar to the linearly separable case, the optimization problem defined in equation (3.8) can be transformed to form the following dual optimization problem:

$$max_{\alpha} L_D = \textstyle\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \,.\, x_j \tag{3.9}$$

Subject to: $\forall_i \begin{cases} \sum_i \alpha_i y_i = 0 \\ C \;\leq \alpha_i \;\leq 0 \end{cases}$

3.1.1.3 Karush-Kuhn-Tucker (KKT) conditions

The SVM optimization problem is a convex problem, hence, given the optimization problem defined in equation (3.6), the following KKT conditions are the necessary and sufficient conditions that must be satisfied for ω^*, b^* and α^* to be a solution [201].

$$\frac{\partial L(\omega^*, b^*, \alpha^*)}{\partial \omega} = \omega_v - \textstyle\sum_i \alpha_i y_i x_{iv} = 0 \quad v = 1, \dots, d \tag{3.10}$$

$$\frac{\partial L(\omega^*, b^*, \alpha^*)}{\partial \omega} = -\sum_i \alpha_i y_i = 0 \tag{3.11}$$

$$y_i(x_i \,.\, \omega + b) - 1 \geq 0, \ \forall_i \tag{3.12}$$

$$\alpha_i \geq 0 \quad \forall_i \tag{3.13}$$

$$\alpha_i(y_i(\omega \,.\, x_i + b) - 1) = 0, \ \forall_i \tag{3.14}$$

3.1.1.4 Dual optimization problem

Solving the primal optimization problem (over α_i), leads to the following SVM formulation, called the dual optimization problem [202]:

$$max_\alpha L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \,.\, x_j \tag{3.15}$$

Subject to: $\forall_i \begin{cases} \sum_i \alpha_i y_i = 0 \\ \alpha_i \geq 0 \end{cases}$

Solving the above dual problem produces some bunch of alpha (α_i) coefficients. Positive alpha coefficients are the most important points for classification, hence they are called support vectors. To save computational time, only instances with positive alpha values are normally used for classification.

3.1.2 Non-linear support vector machine

SVM is not only useful for linear decision boundaries, it can also be extended to handle non-linear decision boundaries. This is achieved by using kernel functions, which map or transform the non-linear data space to a higher dimensional feature space [27]. For non-linear cases, the following optimization problem is solved [29]:

$$max_\alpha L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{3.16}$$

Subject to: $\forall_i \begin{cases} \sum_i \alpha_i y_i = 0 \\ C \leq \alpha_i \leq 0 \end{cases}$

The optimization problem defined in equation (3.15) is similar to the problem defined in equation (3.16). The major difference is in the dot product (i.e. $K(x_i, x_j)$), which is the dot product in the mapped or higher dimensional space. Kernel functions could be valid or invalid. Kernel functions are said to be valid if they satisfy the Mercer condition. The optimization problem for invalid kernel functions may be unsolvable [202]. Some popular kernel functions include [27]:

i. Linear Kernel: $K(x_i, x_j) = x_i^T x_j$

ii. Polynomial Kernel: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$

iii. Radial Basis Function (RBF) Kernel: $K(x_i, x_j) = \exp(-\gamma \parallel x_i - x_j \parallel^2),\ \gamma > 0$

iv. Sigmoid Function kernel: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

where $\gamma, r\ and\ d$ are the kernel parameters.

3.1.3 Support Vector Machine quadratic programming solvers

Optimization problems can be solved using Quadratic Programming (QP) solvers. Solving the optimization problem produces different hyperplanes and the hyperplane with the best margin is usually selected and used for classification. Some SVM training methods utilizes sequential QP solvers while others uses a faster and improved QP solver, called Sequential Minimal Optimization (SMO).

3.1.3.1 Sequential quadratic programming

Sequential QP is a popular QP technique used for solving numerical QP nonlinear optimization problems. Consider the following nonlinear optimization problem:

$$min_x\, f(x) \tag{3.17}$$

Subject to: $d(x) \geq 0$

$$e(x) = 0 \tag{3.18}$$

where $x \in \mathbb{R}^n$

The Langrangian formulation for the above nonlinear optimization problem is given as follows [203]:

$$\mathcal{L}(x, \lambda, \sigma) = f(x) - \lambda^T d(x) - \sigma^T e(x) \tag{3.19}$$

where $\lambda\ and\ \sigma$ represent Lagrange multipliers. Solving equation (3.19) using sequential QP, a suitable search direction can be defined as a solution to the following QP sub-problem:

$$min_g f(x_i) + \nabla f(x_i)^T g + \frac{1}{2} g^T \nabla_{xx}^2 \mathcal{L}(x_i, \lambda_i, \sigma_i) g \tag{3.20}$$

Subject to: $d(x_i) + \nabla e(x_i)^T g \geq 0$

$$e(x_i) + \nabla e(x_i)^T g = 0$$

3.1.1.2 Sequential minimal optimization

SMO, proposed by Platt [204], is an improvement over the existing SVM algorithms, which uses numerical QP. The algorithm was proposed to produce faster solutions to SVM. SMO was designed to solve the minutest optimization problem, which in standard SVM, involves two Lagrange multipliers. At each iteration, SMO selects and optimizes two Lagrange multipliers, and immediately update SVM to show the newly-optimized values. Platt [204] noted that two Lagrange multipliers can be solved using analytical methods, hence, SMO totally avoided the use of numerical QP, which is time consuming. To solve for the two Lagrange multipliers, firstly, SMO computes their constraints and finally computes their minimum. For more information on SMO, the interested reader is directed to [204].

3.1.4 Support vector machine computational complexity

Time complexity for an algorithm measures the total execution time for the algorithm. Time complexity is commonly expressed in terms of Big O notation. Given a matrix $X \in \mathbb{R}^{n*d}$ where n denotes the number of matrix points and d represent the matrix dimension. Computational complexity for the SVM primal optimization problem is $O(nd^2 + d^3)$ [205]. Also, computational complexity for the SVM dual optimization problem is $O(dn^2 + n^3)$ [205]. Chapelle [205] noted that either the primal or dual optimization problem could be solved depending on whether the size of n is less or greater than d. This results in a computational complexity of $O(\max(n,d)\min(n,d)^2)$. Obviously, the computational complexity of SVM is high, making it unrealistic to handle applications that process vast volumes of data. Hence, this thesis proposes different instance selection techniques that reduce the number of training instances (i.e. n), and consequently the computational complexity, without significantly affecting the classification quality.

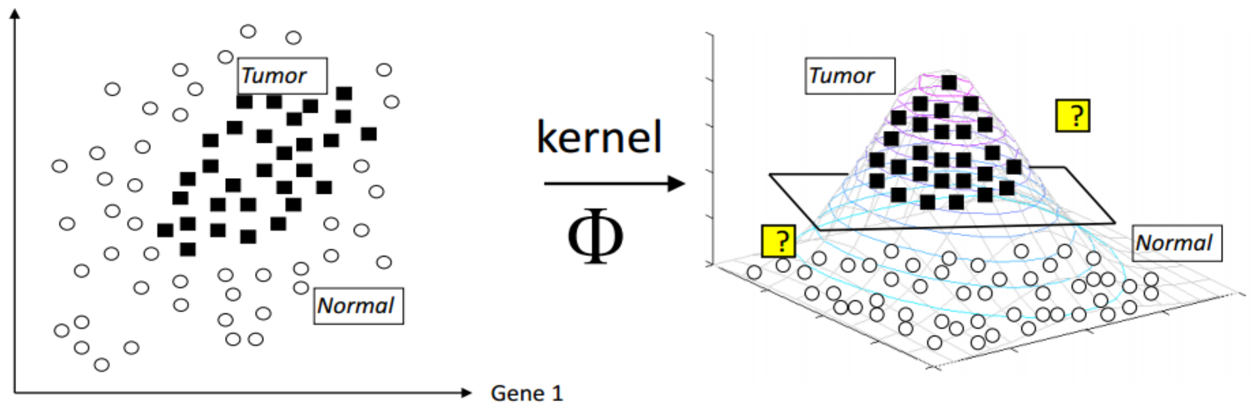

Figure 3.1: Linearly separable vs. non-linearly separable data [202]

3.2 Instance selection preliminaries

Instance-Based Learning (IBL) algorithms make use of the NN classifier for classification. The nearest neighbour algorithm performs classification by searching for nearby instances that have been detected beforehand. Instance-based classifiers comprehensively store training instances [7]. This leads to indiscriminate storage of irrelevant instances [7], which enormously affect their computational complexity. Class structure formed by different instances varies between problems, hence a single instance selection technique cannot be used to effectively handle different problems [7]. This implies that class structure is an important factor that must be considered when developing instance selection techniques [7]. IBL algorithms typically face the challenge of selecting relevant instances that are suitable for classification [206]. Processing a large volume of instances requires large memory space and leads to poor classification speed and noise oversensitivity [206].

Instance selection is performed by using different approaches: competence enhancement and competence preservation [7]. Competence enhancement aims to remove noisy instances from a dataset and consequently increases the accuracy of a classifier [7]. Competence preservation aims to remove superfluous instances. Typically, competence enhancement has a higher probability of improving the classification accuracy since only low number of instances are removed. Competence preservation has a low probability of improving the classification accuracy, since many instances are removed [7]. However, in competence preservation the classification accuracy is preserved.

A typical training set consists of different instances, where each instance consists of input vectors and a predicted output value. Typically, a ML algorithm is trained on a set of instances (called training dataset) and tested on a completely different set of instances (called test dataset). Machine-learning algorithms are expected to predict the output values of the instances in the test dataset. Many ML algorithms perform prediction by using a distance function which computes the distance between learned patterns and incoming instances [206]. Learned patterns could be from the training instances, hyperrectangles, prototypes or rules [206]. The number of instances to store and the size of the instance space are two major challenges faced by IBL algorithms.

IBL algorithms is a subclass of exemplar-based learning algorithms [206]. Other sub-classes include [206]: memory-based learning [207], exemplar-based generalization [208] and case-based reasoning [209]. IBL algorithms use training instances as exemplars [206]. The NN is an example of an IBL algorithm. During classification, IBL algorithms utilize a distance function to compute the proximity of each stored instance to each input instance, and predict the class of the input instance based on its proximity to the stored instances [206]. It is essential to only store relevant instances during classification. Irrelevant instances will negatively affect generalization performance, classification accuracy and speed [206].

3.2.1 Instance space structure

The manner in which classification algorithms detect relevant instances in an instance space is presumed to be similar [7]. Designing a single technique that can solve any classification problem would be the perfect situation. However, this may not be possible because, there are two wide groups of instance space structures, and each of these structures requires different classification approaches [7]. The structures include: homogenous and non-homogenous class structure [7].

3.2.2 Homogenous class structure

Homogenous class structure refers to class structures containing a group of instances with similar neighbourhood [7]. Most of the existing classification problems in ML have a homogeneous class structure [7]. If we have a class with homogeneous instance collection, relevant instances can be recognized by identifying prototype instances or top quality instance. Instances that are on the class borders are always very important to the prediction process. Additionally, instances that are far from the borders (called interior instances) are not relevant to the classification process, hence their removal does not affect the classification accuracy of a NN classifier [7]. Although high utility instances may also be on the class borders, it is not certain that these instances are relevant to the classification process [7]. This is because, identification of high quality instances requires a feedback that shows instances that have been previously used for classification [7]. Instances that have been previously used will likely have an accurate utility value and instances that have not been previously used will likely have an incorrect utility value [7].

3.2.3 Non-homogenous class structure

A non-homogeneous class refers to class structures with different neighbourhood. Instance selection from an instance space with non-homogeneous class structure is not reasonable, because every instance in a non-homogeneous class is a critical instance, since they all belong to the same neighbourhood [7]. Hence, the best way to remove instances from an instance space with non-homogenous class structure is by identifying prototype instances [7].

3.2.4 Instance selection design and search techniques

Instance selection algorithms are generally designed to select a reduced set from a training dataset. They can also be designed to transform instances using different representation techniques such as prototypes [208] and hyperrectangles [210]. Regarding instance selection algorithms that aim to select a reduced set, data points may be absent from the exact points where accurate classification accuracy can be achieved; unlike prototypes and hyperrectangles, which can be designed to be in regions where classification accuracy can be precisely achieved [206]. There are three search methods utilized by instance selection techniques: incremental, decremental and batch search techniques.

3.2.4.1 Incremental search technique

Incremental search technique starts with an empty subset (S), and adds relevant instances into S during the process of selection. The added instances are based on whether they fulfil some user-defined conditions. The order in which the instances are presented matters, because some instances will likely not be included in S if they were visited at a later time. Also, some incremental algorithms do not retain all training instances during the learning phase. Hence, in incremental technique, the presentation order is very important [206]. In an incremental search, more relevant instances can be added to the subset (using the same defined conditions), even after the training phase. Also, an incremental search is faster than non-incremental techniques and some training instances can be discarded during the learning phase. Incremental search requires $O(ns)$ for time and $O(s)$ for storage, instead of $O(n)^2$ for time and $O(n)$ for storage, where n is the number of instances in the training dataset and s is the number of retained instances. One of the main drawbacks of incremental techniques, is their sensitivity to the order of presentation of instances [206]. Also, in the incremental technique, the first few instances are prone to be misclassified, since their classification is based on few presented instances [206].

3.2.4.2 Decremental search technique

The decremental search technique starts with the entire training set ($S = T$), and removes instances from S based on some conditions [6]. The order of presentation of instances is also important in the decremental search technique [6]. However, during the learning phase, all the training instances are not discarded and they are available throughout the phase. Hence, this search technique requires more space, and it is slower than the incremental search technique.

3.2.4.3 Batch search technique

In the batch search technique, the entire dataset is processed at once and irrelevant instances are discarded. Hence, instead of constantly updating a subset, instances that meet a pre-defined condition are removed from the training set at once, and the others are retained for classification. Batch removal may be detrimental, especially if the removal condition is not well defined. The batch search technique also has high computational complexity compared to the incremental technique [206].

3.2.4.4 Instance selection techniques selection criteria

Some instance selection techniques aim to store border instances, because border instances contribute more to the decision surface compared to non-border instances (called interior instances). Removal of non-border instances does not significantly affect a decision surface, hence their removal has a negligible effect on classification accuracy [206]. However, some instance selection techniques remove border instances. That is, noisy instances or instances that disagrees with their neighbours [206]. Removal of these instances result in a better decision surface, but may also affect the decision process, since some irrelevant instances are retained [206].

Selecting the suitable number of nearest neighbours (i.e. k) is a challenging task in instance selection [206]. It is important that the value of k is set to an odd integer number, so that the votes for the majority class will always be greater than the votes for the minority class. This will also ensure that an input vector does not have two predicted classes. Additionally, it is important that the value of k is always greater than one, to ensure that new instances are not always assigned to one class. Leave-One-Out cross-validation is one of the popular methods that can be used to select the value of k [206]. In Leave-One-Out cross validation, each value of k is evaluated by classifying each instance by its k nearest neighbours, leaving out the instance itself [206]. Afterwards, the k value that yields the best classification accuracy is selected [206]. Some algorithms, such as RBF, use a weighting scheme that permits every instance in the dataset to vote.

3.2.5 Types of instance selection

Similar to feature selection, instance selection techniques can be grouped into two areas: wrapper and filter. The major difference between wrapper and filter instance selection techniques is in their method of selection. Wrapper-based instance selection techniques utilize a classifier to evaluate the accuracy of each subset during the selection phase [6]. As shown in Figure 3.2, instances that contribute less to classification accuracy are removed from the dataset [6]. Filter-based instance selection techniques do not depend on the accuracy of a classifier; instances are selected based on a fitness function [6]. Instances with low fitness value are removed from the training dataset and instances with high fitness value are retained in the dataset. Filter-based instance selection techniques are generally faster than wrapper-based instance techniques, especially, when a large volume of data processing is involved [6].

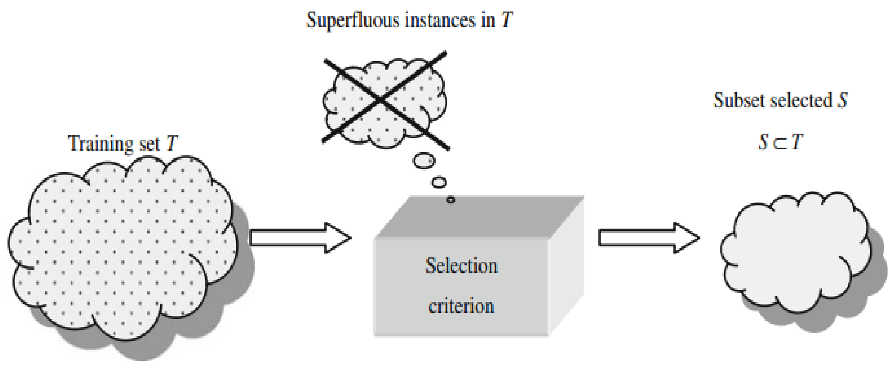

Figure 3.2: Instance selection process
Source: [6]

3.3 Boundary detection algorithms

As aforementioned, two of the proposed filter-based techniques perform two main actions: boundary detection and instance selection. Firstly, they detect a boundary, and then select instances close to the boundary. The two boundary detection techniques are discussed next.

3.3.1 Edge Detection instance selection algorithm

This thesis proposes an instance selection technique called Edge Detection Instance Selection Algorithm (EDISA). EDISA is a boundary detection algorithm, inspired by edge detection in image processing. A brief introduction to edge detection is presented next, followed by details on the proposed edge detection algorithm.

3.3.1.1 Edge detection overview

Edge detection in image processing is a technique used to identify object boundaries in images [211]. Object boundaries are points in images with sharp changes in image brightness [211]. Generally, images contain some quantity of redundant data that requires pruning for effective classification. Hence, to reduce computational complexity, edge detection is a highly essential pre-processing step [212]. Edge detection is applied to images with the aim of identifying important features, removing less-relevant information and consequently reducing the image size. Generally, edge detection is used for segmentation of images, feature extraction, and feature detection in image processing, computer vision and machine vision [211-213]. Edge detection conserves essential structural properties of images and computer space [212]. Some edge detection algorithms include: Canny algorithm, Sobel algorithm, Roberts algorithm, etc. Figure 3.3 shows an example of an image and its detected edges.

The concept of Edge detection in image processing is similar to the concept of boundary detection in SVM classification. Edge detection aims to select objects located at boundary positions, and boundary detection algorithms aims to select instances (also called support vectors) close to a decision boundary. In this research, an instance selection technique that is inspired by edge detection is proposed.

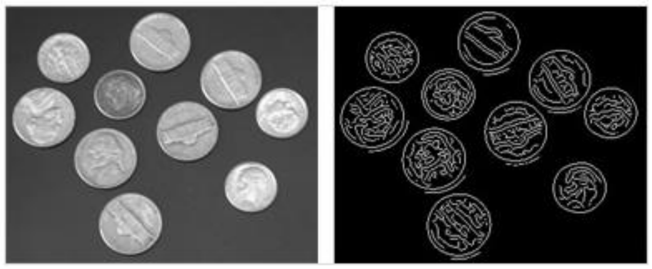

Figure 3.3: Example of edge detection [211]

3.3.1.2 Edge detection instance selection algorithm

As aforementioned, EDISA is inspired by edge detection in image processing. Given a set of training instances, EDISA identifies an edge instance and selects N instances close to it. As shown in Algorithm 3.1, the algorithm begins by initializing the vote count for all instances in the dataset (line 1). The vote count shows the number of times each instance is voted (as an edge instance) by other instances. To enhance speed, the filter-based techniques are designed to utilize just a fraction of the entire dataset, hence, in line 3, the algorithm randomly select M instances from the training dataset (line 3). Furthermore, for each $\mathrm{instance_j}$ in the dataset, EDISA computes their individual neighbourhood list by computing the squared Euclidian distance between $\mathrm{instance_j}$ and other instances in the dataset (line 6). In addition, for each $instance_j$, based on the proximity of other instances in the neighbourhood list of $\mathrm{instance_j}$, EDISA votes a corresponding edge instance, $instance_k$ (line 8). $\mathrm{Instance_k}$ is the edge instance for $\mathrm{instance_j}$, if it is the farthest instance from $\mathrm{instance_j}$. That is, $\mathrm{instance_k}$ is the edge instance of $\mathrm{instance_j}$, if it has the highest euclidean distance compared to other instances in the neighbourhood list of $\mathrm{instance_j}$. Furthermore, in line 12, EDISA increases the vote count for the current voted edge instance ($instance_k$). The process is repeated until the neighbourhood list of all the instances in the dataset has been processed. Afterwards, EDISA selects the instance with the majority (or highest) vote (line 14), and computes the k nearest neighbours to the selected instance (i.e. the selected edge). Finally, the identified k nearest neighbours are used to train SVM. Some experiments were performed to test the efficiency of EDISA, and the result reveals that EDISA significantly improved the SVM classification speed. Experimental results are presented in Section 4.4.

Algorithm 3.1: Edge Instance Selection Algorithm

```
Input: $N, Nsub, K$
Output: EI[] /* output array of edge instances for training */
1 Initialize $Vote[N]$ /* Initialize vote count for each instance */
2 $DECLARE\ farthest, dist[,], index$
3 Randomly select M instances from dataset, where M = Nsub
4 For $j$ = 1 to $N$
5     For $k$ = 1 to $N$
6         $dist[j,k] = SquaredEuclideanDistance(instance_j, instance_k), where\ j \neq k$
7         if $dist[j,k] > farthest$
8             $farthest \leftarrow dist[j,k]$ /*get the farthest $instance_k$ from $instance_j$*/
9             $index \leftarrow k$ /*save the index of the farthest $instance_k$ */
10        end if
11    End $k$
12    $Vote[index] += 1$ /*Increase vote count for the farthest $instance_k$*/
13 End $j$
14 $E \leftarrow Vote.Max()$ /*Select the edge, i.e. instance with the majority vote */
15 EI $\leftarrow ComputeKNN(E)$ /*Get edge instances, i.e. k nearest neighbors from E*/
16 Return EI
```

3.3.2 ACO inspired boundary detection and instance selection technique

This thesis proposes a boundary detection and instance selection technique, called Ant Colony Optimization Instance Selection Algorithm (ACOISA). As shown in Figure 3.4, ants move randomly in search for food, and immediately an ant locates a food source, it deposits pheromone trails on its way back to its nest. The deposited pheromone trails leads other ants to the same food source, thereby minimizing the total time taken for a food search. Trails with the highest pheromone concentration lead to the best food source. Inspired by the same concept, ACOISA searches a dataset for boundary instances, and selects the instance with the highest pheromone value. The instance with the highest pheromone value is the best boundary instance. Subsequently, n instances close to the selected boundary instance are selected and used for training, where n is user defined. It is worth mentioning that in ACOISA, ACO algorithm was primarily used for boundary identification and not instance selection. A full description of ACOISA is presented in Section 3.3.2.2.

3.3.2.1 Ant Colony Optimization overview

ACO is a swarm intelligence-based algorithm, inspired by the foraging behaviour of ant colonies. The initial algorithm, originally proposed by Marco Dorigo [214], seeks to find the best path in a graph. The idea was motivated by ant behaviours, which seek to search for the best path between their colony and a food source. In their search for food sources, ants randomly move around the region surrounding their colony [215]. Upon locating a food source (as shown in Figure 3.4), ants take some of the food back to their nest, and simultaneously deposit pheromones on the ground while returning [215]. This deposited pheromones, forms a trail and guides other colony members to the food source. Other colony members also deposit pheromones on the ground, when returning to their colony. Hence, paths frequently walked upon by ants form the shortest path and thus influence an ant's choice of path [216]. Generally, ants use pheromones as a means of communication. Ant movements are controlled by a probabilistic action rule. Movement of a given ant, from one node (node a) to another node (node b), is determined by equation (3.21) [217].

$$p_{i,j}^{(n)} = \frac{\left(\tau_{a,b}^{n-1}\right)^{\alpha}\left(\eta_{a,b}\right)^{\beta}}{\sum_{j \in \Omega_a}\left(\tau_{a,b}^{n-1}\right)^{\alpha}\left(\eta_{a,b}\right)^{\beta}},\ if\ b\ \in \Omega_a \tag{3.21}$$

where $\tau_{a,b}^{n-1}$ is the pheromone deposited on the arc connecting $node\ a$ to $node\ b$, Ω_a is the set of possible nodes an ant can visit, given that the ant is on node a; α controls the impact of pheromone information, β controls the heuristic information. $\alpha\ and\ \beta$ are constants. $\eta_{a,b}$ is the heuristic information for an ant moving from node a to node b, and it is a fixed value for each iteration. In ACO, pheromone matrix is updated twice. The first update is performed after movement of each ant within each construction step, and the second update is performed after all the ants have moved within each construction step [217]. The first and second updates are represented by equations (3.22) and (3.23), respectively [217].

$$\tau_{a,b}^{n-1} = \begin{cases} (1-\rho) * \tau_{a,b}^{n-1} + \rho * \Delta_{a,b}^{(k)}, & if\ (a,b)\ is\ currently\ visited \\ \tau_{a,b}^{n-1}, & otherwise \end{cases} \tag{3.22}$$

where ρ is the evaporation rate.

$$\tau^{(n)} = (1-\psi) * \tau^{(n-1)} + \psi * \tau^{(0)}, \tag{3.23}$$

where ψ is the pheromone decay coefficient.

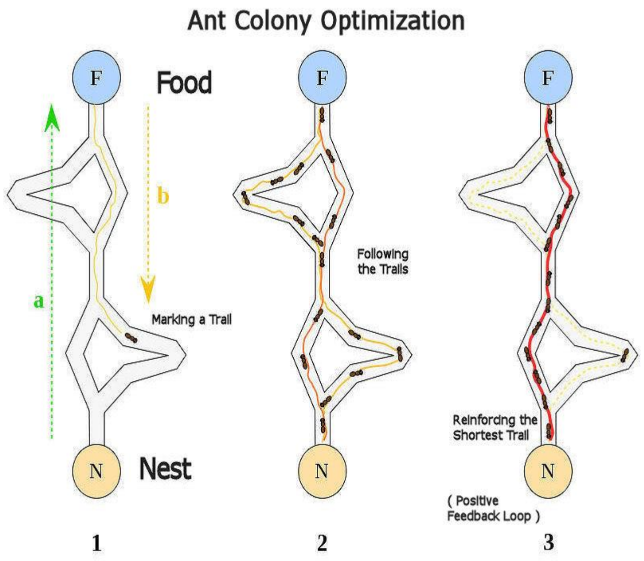

Figure 3.4: Description of ant colony optimization [218]

3.3.2.2 Ant Colony Instance Selection Algorithm

ACOISA is inspired by edge selection in ACO, and it uses ACO to construct a pheromone matrix, where each entry in the matrix represent the boundary information for each instance in the dataset. As shown in Algorithm 3.2, ACOISA begins by randomly selecting M subset of instances from the dataset, where M is user-defined (line 8). Furthermore, N ants are randomly assigned to all the instances in the subset, where each instance represents a node (line 9). Further, ACOISA computes the heuristic value for all instances in the dataset (line 10 – 21). As aforementioned, ACOISA aims to select the boundary instance with the highest pheromone value. Hence, the heuristic value for each instance is designed to reflect the boundary information of each instance. ACOISA computes the heuristic value for $instance_i$ by first computing the Euclidean distance between $instance_i$ and other instances in the data subset (line 11-14). Afterwards, based on the computed distances, ACOISA selects the k nearest neighbours to $instance_i$ (line 15). Furthermore, ACOISA selects all the instances of the opposite class in the neighbourhood list of $instance_i$ and sum their individual distances (line 16-20). For example, if there are two classes ($class\ a$ and $class\ b$), and $instance_i$ belongs to $class\ a$, the computed Euclidean distances between $instance_i$ and other instances in its neighbourhood list, belonging to $class\ b$, will be summed and used as the heuristic value for $instance_i$. This process is repeated until the heuristic value for all the instances in the dataset are computed. At the end of the heuristic value computation stage, the instance with the highest heuristic value, will be the instance with the largest number of selected boundary instances, and the instance that contain boundary instances that are far away from their respective boundaries. The farther they are from the boundary, the wider the margin. And, recall that the goal of SVM is to select the boundary with the largest margin. Hence, the instance with the highest heuristic value will be the instance with the widest margin.

After the heuristic value for all instances has been computed, N ants are moved around the dataset (line 22-31). Initially, an ant is randomly selected and moved for a pre-defined number of steps (line 24-27). This ant moves from one node ($node\ x$) to another node ($node\ y$) in its neighborhood list, according to a probability defined in equation (3.24).

$$p_{x,y}^{(n)} = \frac{(\tau_y^{n-1})^{\alpha}\,(\eta_y)^{\beta}}{\sum_{y\,\epsilon\,\Omega_x}(\tau_y^{n-1})^{\alpha}\,(\eta_y)^{\beta}}, \tag{3.24}$$

where τ_y^{n-1} refers to the pheromone value of $node\ y$, Ω_x is the neighbourhood list of $node\ x$, η_y represent the heuristic information at node y. $\alpha\ and\ \beta$ are user-defined constants. α controls the pheromone matrix and β controls the heuristic matrix. Each node has five values: initial pheromone value, current pheromone value, heuristic value, position and neighbourhood nodes. Initial pheromone value is set to a constant value, heuristic value is computed using k-NN, as explained above. The position for each ant is defined by equation (3.24), and neighbourhood nodes for each node contain the list of possible nodes that a given ant can move to. The current pheromone value is updated twice. The first update is performed in line 26, according to equation (3.25). This update is performed every time an ant is moved. The second update is performed after all ants have been moved, according to equation (3.25) (line 29).

$$\tau_y^{n-1} = \begin{cases} (1-\rho) * \tau_y^{n-1} + \rho * \Delta_y^{(z)}, & if\ node\ y\ is\ currently\ visted \\ \tau_y^{n-1}, & otherwise \end{cases} \tag{3.25}$$

where ρ is the evaporation rate, $\Delta_y^{(z)}$represent the heuristic information at the zth node. Finally, the best node (node with the highest pheromone value) is selected in line 32, and k-NN algorithm is used to select boundary instances (line 33). Boundary instances refer to instances close to the selected best node. Finally, the selected boundary instances are used to train SVM (line 35) and the average predictive accuracy is computed. This process is repeated until N runs have been reached, where N is user defined.

Algorithm 3.2: Ant Colony Optimization Instance Selection Algorithm

Input: $D, NF, MaxG, N, NSub, NR, NRuns, K$

Output: ACA

```
1 Start SVM /* main method /*
2 for $i$ = 1 to $NRuns$
3   for $j$ = 1 to $NF$
4       $TrainingDataset \leftarrow 9/10\ of\ D$ /*Get the training dataset for the current fold */
5       $TestDataset \leftarrow 1/10\ of\ D$ /* Get the test dataset for the current fold */
6       $ACOISA(TrainingDataset)$ /*Pass training subset to ACOISA for boundary and instance selection*/
7           Randomly select M training instances from TrainingDataset, where M = size of training subset
8           Randomly assign ants to instances and initialize pheromone value for all ants
9           for $a$ = 1 to $N$ /* Compute heuristic value for all instances in dataset */
10              for $b$ = 1 to $N$
11                  Compute distance between $\text{instance}_a$ and $\text{instance}_b$, where $a \neq b$
12                  dist[a, b] = distance
13              end b
14              NL[a] ← $ComputeKNN(dist[a])$ /*Compute k nearest neighbours for $instance_a$ */
15              for $b$ = 1 to $K$ /* Compute heuristic value for each instance */
16                  if Class of $\text{instance}_a \neq$ Class of NL[a, b]
17                      $HV[a] +=$ dist[a, b] /*compute the heuristic value for $instance_a$*/
18                  end if
19              end b
20          end a
21          while ($p < MaxG$) /* Start moving ants */
22              for $k$ = 1 to $N$
23                  for $m$ = 1 to $NR$
24                      Move $k^{th}$ ant to $m^{th}$ neighbouring node using equation (3.24)
25                      Perform update using equation (3.25)
26                  end $m$
27              end $k$
28              Perform update using equation (3.23)
29              $p{+}{+}$
30          end while
31          E ← $SelectBoundaryInstance(HV, D)$ /*Select instance with the highest heuristic value*/
32          $EI \leftarrow ComputeKNN(E)$ /*Select k nearest neighbours to the Boundary Instance*/
33      end ACOISA
34      TrainSVM(EI) /*Train SVM on the selected Edge instances*/
35      $newCA \leftarrow evaluateModel(TestSubset)$
36      $CA += newCA$
37  end $j$
38  $ACA += CA / NF$
39 end $i$
40 Output $ACA/NRuns$
41 end SVM
```

3.4 Nature inspired instance selection algorithms

Nature solves complex problems of varying difficulties. Interestingly, nature solves these complex problems in simple ways. This has inspired many researchers to design algorithms that are useful for solving real-world complex problems, such as e-fraud detection [219], travelling salesman problem [220], parking lot problem [2] and hostel allocation problems [3]. These algorithms are called NI algorithms. In simple terms, NI techniques refer to algorithms that are inspired by the problem-solving ability of nature.

Examples of NI algorithms include: river formation dynamics algorithm [221], simulated annealing [222], FPA [223] and SSA [224]. River formation dynamics is inspired by the manner in which water drops form a river bed; simulated annealing is inspired by the annealing process of metals; FPA is inspired by the pollination process of flowers and SSA is inspired by the foraging behaviour of social spiders. NI algorithms can be used to handle problems without relying on existing domain knowledge [25]. They can also be used to handle problems in the presence of noise or outliers [25]. Moreover, unlike many AI techniques, NI algorithms are robust and dynamic, they can easily adjust to a fluctuating environment [25], such as e-fraud detection. NI algorithms are designed to handle continuous problems, nevertheless, since instance selection is a binary problem, as shown by Jordehi and Jasni [225], the sigmoid function or the rounding off approach can be used to convert each agent position from continuous to binary values (0 or 1), where 1 indicates that an instance is selected, and 0 indicate that an instance is not selected. In this research, some experiments were performed to check the effectiveness of the Rounding off approach and the sigmoid function. Experimental results show that, for FPA and FFA, the values from the sigmoid function are mostly skewed towards 1, while the values from the rounding off approach are uniformly distributed. Moreover, for SSA, CSA and BA, the values from the sigmoid function are uniformly distributed. Hence, in this study, the rounding off approach is used to convert each flower and firefly position from continuous to binary values (0 or 1). Also, the sigmoid function is used to convert each spider, cuckoo and bat position from continuous to binary values. The sigmoid function used in this research is given in equation (3.41). Also, the rounding function used in this research is given in equation (3.43).

Given N training instances, utilizing the entire training set for training is time consuming. Brighton and Mellish [7] noted that 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. On this basis, all the proposed filter-based techniques were designed to use only a subset of the entire training set for instance selection. That is, for all experiments, n instances were passed to the instance selection techniques for processing, where $n < N$. This implies that each technique searches an instance space consisting of n instances, instead of an instance space consisting of N instances (i.e. the entire training dataset). Also, for all the experiments, different sets of parameters were evaluated, with the aim of determining the best parameters that are suitable for all the proposed techniques and also demonstrating the robustness of the proposed techniques. Unlike the filter-based techniques, the wrapper-based techniques are designed to explore the entire training set. That is, the proposed wrapper-based techniques are designed to search through the entire N training instances for relevant solutions.

This thesis proposes five filter-based and wrapper-based instance selection techniques. The proposed techniques are called: CSISA, BISA, FPISA, SSISA and FFISA. The major difference between the filter and wrapper based techniques is in their method of selection. The filter-based techniques rely on a fitness function for instance selection, while the wrapper-based techniques rely on SVM algorithm for instance selection. The filter-based techniques aims to improve the training speed of SVM, and the wrapper-based techniques aims to improve the predictive accuracy of SVM. Pseudocode for the filter and wrapper based techniques is shown in Algorithms 3.1 – 3.7. The flowchart for the wrapper and filter based algorithms is shown in Figure 3.5.

The filter-based techniques are designed with the objective of maximizing percentage reduction and boundary instances used for training. The wrapper-based techniques are designed with the objective of maximizing the training speed and predictive accuracy of SVM. Both techniques use different agents to search for relevant instances, where each agent consists of a binary array of N instances, called an instance mask. Given a set of training instances, each algorithm starts by randomly initializing the instance mask for each agent to 0 and 1, where 1 indicates that an instance is selected, and 0 indicates otherwise. Afterwards, the fitness value for each agent is computed and the global best is saved. Fitness function for the filter-based and wrapper-based techniques are reported in Sections 3.5.1 and 3.5.2, respectively. Furthermore, new solutions are generated at different iterations and the global best solution is selected at the end of the final iteration. Finally, the agent with the best solution is selected and used to train SVM. A constraint is added to ensure that at least N instances are selected for training, where N is user-defined. Hence, if the total number of selected instances (S) is less than N, P additional instances are randomly selected, where $P = N - S$. This constraint is added to eliminate the possibility of having zero selected instances. More details on the five algorithms are presented next.

3.4.1 Cuckoo search-inspired technique

This thesis proposes a NI-based instance selection technique called CSISA. Two different variants of this technique are proposed in this thesis. The first variant (filter-based) is designed to improve the training speed of SVM and the second variant (wrapper-based) is designed to improve the predictive accuracy of SVM. More details on CSISA are reported in this section.

3.4.1.1 Cuckoo search algorithm overview

The CSA, proposed by Yang [226], is inspired by the parasitic behaviour of some species of cuckoo birds, and the levy flight behaviour of some fruit flies and birds species. The reproduction strategy of cuckoo birds is violent. Some species rely on other birds to hatch eggs and feed their young. These species (called brood parasites) lay their eggs in nests of other birds [226]. Mostly, they target nests of birds that have newly laid their eggs.

Generally, cuckoo eggs hatch earlier than their host eggs, hence, by instinct, the newly hatched cuckoo throws the host eggs out their nest, to increase the share of food provided by the host bird [226]. CSA was developed based on this parasitic behaviour of cuckoos. The following idealized rules were used to develop CSA:

i. Each cuckoo lays one egg at a time, and randomly distributes its egg to different nests
ii. The best nest, containing high quality eggs, will survive to the next generation
iii. The number of host nests is fixed. Also, eggs laid by a cuckoo are discovered by the host bird by a probability of $p_a \in [0, 1]$. If eggs are discovered, the host bird can either abandon its nest and build a new nest, or tip the discovered eggs out of the nest.

New positions for each cuckoo are generated by performing a levy flight, given in equation (3.26).

$$X_i^{(t-1)} = X_i^{(t)} + \alpha \oplus Levy\,(\lambda), \tag{3.26}$$

Yang [226] noted that $\alpha = 1$ is mostly used. $\alpha > 0$ 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 (3.27). The levy distribution has an infinite variance and infinite mean.

$$Levy \sim u = t^{-\lambda}, \qquad (1 < \lambda \leq 3) \tag{3.27}$$

The CSA was originally designed for the continuous problem. However, in this study, the sigmoid function (shown in equation (3.28)) is used to convert each cuckoo position to a binary value (0 or 1). One indicates that an instance is selected, and zero indicates otherwise.

$$S(V_i^t) = \frac{1}{1+ e^{-V_i^t}}, \tag{3.28}$$

Hence, in place of equation (3.26), the position of each cuckoo is updated by equation (3.29):

$$X_i^t = \begin{cases} 1 \; if \; rand() \leq S(V_i^t), \\ 0 \quad otherwise, \end{cases} \tag{3.29}$$

where $rand()$ is a random number uniformly drawn from the range [0, 1]. This study proposes an instance selection algorithm based on standard CSA [226].

3.4.1.2 Cuckoo Search Instance Selection Algorithm

CSISA is a NI-based instance selection algorithm, inspired by CSA. As shown in Algorithm 3.3, the algorithm starts by randomly selecting M instances from the training dataset for training, where M is the size of the training subset to be explored by CSISA (line 6). The algorithm proceeds by initializing the positions for each nest, and also setting other parameters, including Min, where Min is the minimum number of instances to be selected for training (line 10). Each nest position is initialized to 0 or 1, where 1 indicates that an instance is selected and 0 indicates that an instance is not selected. Furthermore, the fitness values for the initialized solutions are calculated and the current best solution is retained (line 14-15). Fitness function for the filter-based and wrapper-based CSISA are described in Sections 3.5.1 and 3.5.2, respectively. Furthermore, new solutions are constructed by randomly selecting different cuckoos through levy flight (line 17-20). The value of each new solution is continuous, hence, they are converted back to binary values using the sigmoid function (line 19). In the real world, if the egg of a cuckoo bird is similar to the eggs of its host, then the egg will unlikely be discovered by the host bird. Otherwise, if the egg is not similar to the host eggs, the discovery rate is higher; the nest will either be abandoned by the host bird, or the egg will be tipped out of the nest. Therefore, the fitness of each nest is related to the solutions they produce. To handle this scenario, new solutions are generated and low quality nests are replaced (line 21-24). Low quality nests are discovered with a user-defined probability (or discovery rate). Furthermore, the quality (or fitness) of the new solutions are evaluated and the global best solution is retained (line 25-26). This process is repeated until it reaches a user-defined number of iterations (line 31) or a user-defined fitness threshold (line 27-29). After the algorithm terminates, the instances selected by the global best solution are used to train SVM. Prior to training, if N instances, selected by the global best solution, is less than a user-defined threshold (Min), then J additional instances are randomly selected from the training dataset and added to the solution space of the global best cuckoo, where $J = N - Min$ (line 33-35). Finally, the instances selected by the global best solution are used to train SVM.

3.4.2 Bat-inspired technique

In this thesis, two different variant of BISA are introduced. Both variants are designed to remove irrelevant instances from datasets and consequently improve the training speed and predictive accuracy of SVM. Experiments were performed to evaluate the efficacy of the two variants and the results show that the first variant (filter-based) produces very fast classification models and the second variant (wrapper-based) produces fast and accurate classification models. More details on BISA are provided in Section 3.4.2.2. A brief introduction to the BA is presented next.

3.4.2.1 Bat Algorithm overview

The BA is inspired by the echolocation behaviour of bats. Most bats uses echolocation to locate food (or preys), avoid obstacles and locate their roost in the dark [227]. Bats emits loud sounds in patterns, and they pay attention to the echo that may reflect back from objects in the surroundings [227]. During hunting, bats emit pulses at a very high rate. However, the rate reduces as they fly closer to a prey [227]. Some bats have good vision, and some have very good smelling ability [227]. This enhances their ability to efficiently detect prey and avoid obstacles [227]. This study proposes an instance-selection algorithm based on the standard BA proposed by Yang [227].

BA was formulated using the following rules [227]:

i. All bats use echolocation to detect distance, and they can differentiate between prey and obstacles
ii. 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 prey. Depending on their target proximity, bats can regulate their rate of pulse emission, and the wavelength of their emitted pulses.
iii. Loudness varies from a large positive value, A_o, to a minimum value, A_{min}.
iv. The position x_i, velocity v_i and frequency f_i for each virtual bat are, firstly, initialized. Furthermore, they are updated as follows [227]:

$$f_i = f_{min} + (f_{max} - f_{min})\,\beta, \tag{3.30}$$

$$V_i^t = V_i^{t-1} + \left(X_i^t - X_*\right) f_i, \tag{3.31}$$

$$X_i^t = X_i^{t-1} + V_i^t \tag{3.32}$$

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 random frequency, randomly selected from $[f_{min}, f_{max}]$. Yang [227] noted that, as soon as a solution is selected from the current best solutions, a new solution is generated locally for each virtual bat in the population, using random walks:

$$X_{new} = X_{old} + \in A^t \tag{3.33}$$

where $\in$ is a random number generated between [-1, 1], and A^t is the loudness of all the bats at every time interval.

Furthermore, per iteration, the loudness and pulse rate emission are regulated as follows:

$$A_i^{t+1} = \propto A_i^t, \tag{3.34}$$

$$r_i^{t+1} = r_i^0[1 - exp(-\gamma t)] \tag{3.35}$$

where $\propto$ *and* γ are BA parameters. Yang [228] noted that $\propto$ can be equal to γ.

Yang [228] also noted that each bat should be randomly assigned a varied pulse emission rate and loudness. Additionally, Yang [227] suggested that Initial loudness A_i^0 can be randomly selected from the range [1, 2], and initial emission r_i^0 rate can be drawn from the range [0, 1]. The original BA was proposed for continuous problems. Each virtual bat moves in continuous space. However, in instance selection, each bat moves in a binary search space, where 1 indicates that an instance is selected and 0 indicates otherwise. In this study, the sigmoid function defined in equation (3.36) is used to convert bat positions to binary values.

$$S(V_i^t) = \frac{1}{1+ e^{-V_i^t}}, \tag{3.36}$$

Hence, in place of equation (3.32), the position of each bat is updated by equation (3.37):

$$X_i^t = \begin{cases} 1 \; if \; rand() \leq \; S(V_i^t), \\ 0 \quad otherwise, \end{cases}, \tag{3.37}$$

where $rand()$ is a random number selected from the range, $[0, 1]$, x_i^k and v_i^k refers to different bat positiond and velocity at different iterations, K refers to dimension.

Algorithm 3.3: Cuckoo Search Instance Selection Algorithm

```
Input: $NF, NI, MaxG, N, NSub\ Min, FT$
Output: ACA
1 Start SVM /* main method */
2 for i = 1 to NRuns
3     for j = 1 to NF
4         $TrainingDataset \leftarrow 9/10\ of\ D$ataset /*Get the training dataset for the current fold */
5         $TestDataset \leftarrow 1/10\ of\ D$ataset /*Get the test dataset for the current fold*/
6         TrainingSubset ← RandomSelect(TrainingDataset) /*randomly select training subset*/
7         CSISA(TrainingSubset) /*Start instance selection*/
8             Define $G(x)$ for cuckoo nests /*define objective function for filter and wrapper-based CSISA*/
9             Initialize Parameters /*initialize all parameter values*/
10            for $a$ = 1 to $N$
11                Initialize solution for $nest_a$ /*initialize solution for all the cuckoo nests*/
12            end for
13            Evaluate $G(x)$ and select $CB$ /*Evaluate the objective function for all cuckoos and select the current best*/
14            $GB \leftarrow CB$ /*Save the current best solution*/
15            while ($p < MaxG$) /*start searching for relevant instances*/
16                for $k = 1\ to\ N$
17                    Construct new solutions by randomly selecting cuckoos using equation (3.26)
18                    Convert new solutions to binary using equation (3.41)
19                end $k$
20                for $a$ = 1 to $N$
21                    Replace low quality nest by generating new solutions using a user-defined probability
22                    Convert new solutions to binary using equation (3.41)
23                end $a$
24                Evaluate $G(x)$ for all new solutions /*Evaluate objective function for the current solutions*/
25                $GB \leftarrow CB$ /*Update the global best with the current best solution*/
26                if $GB > FT$
27                    end while /*Stop algorithm if global best is greater than a pre-defined fitness threshold*/
28                end if
29                $p{+}{+}$
30            end while
31            $NS \leftarrow GB$
32            if $NS < Min$ /*Add more instances if the selected instances are less than a user-defined threshold*/
33                AddInstances(GB) /*Add (Min-NS) instances to the instances selected by the global best*/
34            end if
35            Output $GB$ /*Output the global best solution*/
36        end CSISA
37        TrainSVM(GB) /*Train SVM on the solution selected by CSISA*/
38        $CA += evaluateModel(TestDataset)$ /*Test the trained model on the current test dataset*/
39    end j
40    $ACA += CA\ /\ NF$ /*Add the predictive accuracy produced by the current fold*/
41 end i
42 Output $ACA/NRuns$ /*Output the overall predictive accuracy after all the runs*/
43 end SVM
```

3.4.2.2 Bat Instance Selection Algorithm

Similar to BA, BISA is inspired by the echolocation of bats. The algorithm is shown in Algorithm 3.4. BISA begins by randomly selecting the subset of training instances to be processed (line 7). Plus, the algorithm continues by initializing the parameter values, 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 (line 10 to 15). Furthermore, the fitness value for the initialized solution is calculated and the best solution is retained (line 16). The filter-based BISA uses the fitness function described in Section 3.5 and the wrapper-based BISA uses the fitness function described in Section 3.5.2. Furthermore, the algorithm enters a while-loop to search the dataset for new solutions (line 18 to 39). Within the loop, BISA searches for new solutions by randomly moving within the solution space with velocity and frequency as defined in equations (3.31) and (3.30), respectively (lines 18 to 27). Also, a random number is generated and new solutions are randomly constructed if the random number is greater than a pre-defined pulse rate (line 23 to 26). Furthermore, a bat randomly moves within the solution space and generates new solutions. The new solutions are evaluated and retained if they are better than the previous solution and if a randomly-generated number is less than the user-defined loudness. Further, each of the new solutions is evaluated and the global best solution is updated if a better solution is found. This process is repeated until a user-defined threshold is reached or until solutions converge. Finally, the instances selected by the best solution are used to train SVM (line 46). If the selected instances are less than a user-defined threshold, more instances are added to the selected instances before training SVM. This is to ensure that the algorithm always selects a minimum amount of instances for training. The BA works in a continuous space, hence BISA uses he sigmoid function to convert continuous values to binary values.

Algorithm 3.4: Bat Instance Selection Algorithm

```
Input: $NF, NI, MaxG, N$, NRuns, $Min, D, FT$
Output: $ACA$
1 Start SVM
2 for i = 1 to NRuns
3     for j = 1 to $NF$
4         $TrainingDataset \leftarrow 9/10\ of\ D$ataset /*Get the training dataset for the current fold */
5         $TestDataset \leftarrow 1/10\ of\ D$ataset /*Get the test dataset for the current fold*/
6         TrainingSubset ← RandomSelect(TrainingDataset) /*randomly select training subset from TrainingDataset*/
7         BISA(TrainingSubset) /*/*Start instance selection*/
8             Define $G(x)$ for bats /*define objective function for filter and wrapper-based BISA*/
9             Initialize Parameters /*initialize all parameter values*/
10            for $a$ = 1 to $N$
11                Initialize solution for $bat_a$ /*initialize the solution for each bat*/
12                Define $pr_a$ for $bat_a$ /*specify pulse rate for each bat*/
13                Define $l_a$ for $bat_a$ /*specify loudness for each bat*/
14            end for
15            Evaluate $G(x)$ and select $CB$ /*Evaluate the objective function for all solutions and select the current best*/
16            $GB \leftarrow CB$ /*Save the current best solution*/
17            while ($p < MaxG$) /*start searching for improved solutions*/
18                for $k$ = 1 to $N$
19                    Construct new frequency for $bat_k$ by using $f_{min} + (f_{max} - f_{min})\ \beta$
20                    Construct new velocity for $bat_k$ using $V_k^t = V_k^{t-1} + (X_k^t - X_*)\ f_k$
21                    $R$ ← RandomNumber() /*generate random number between 0 and 1*/
22                    if $R > pr_k$ /*generate a local solution using random walks*/
23                        Construct a solution around $GB$
24                    end if
25                    Convert $bat_k$ to binary using sigmoid function
26                end $k$
27                for $a$ = 1 to $N$
28                    $R$ ← RandomNumber() /*generate random number between 0 and 1*/
29                    Evaluate $G(x_a)$ for new solution /*evaluate the new solutions*/
30                    Replace $solution_a$ if new solution it is better, & if $R < l_a$
31                end $a$
32                Evaluate $G(x)$ for all new solutions /*Evaluate objective function for the current solutions*/
33                $GB \leftarrow CB$ /*Update the global best with the current best solution*/
34                if $GB > FT$
35                    end while /*Stop algorithm if global best is greater than a pre-defined fitness threshold*/
36                end if
37                $p$++
38            end while
39            $NS \leftarrow GB$
40            if $NS < Min$ /*Add more instances if the selected instances are less than a user-defined threshold*/
41                AddInstances(GB) /*Add (Min-NS) instances to the instances selected by the global best*/
42            end if
43            Output $GB$ /*Output the global best solution*/
44        end BISA
45        TrainSVM(GB) /*Train SVM on the solution selected by BISA*/
46        $CA{+}{=}\ evaluateModel(TestDataset)$ /*Test the trained model on the current test dataset*/
47    end j
48    $ACA{+}{=}\ CA\ /\ NF$ /*Add the predictive accuracy produced by the current fold*/
49 end i
50 $ACA \leftarrow CA\ /\ NF$
51 Output ACA / NRuns
52 end SVM
```

3.4.3 Flower pollination-inspired technique

The fascinating pollinating process of flowering plants has attracted the attention of researchers and subsequently inspired them to develop intelligent solutions for optimization problems. This thesis proposes a speed optimization technique based on FPA. More details on the standard FPA and the proposed technique are provided in this section.

3.4.3.1 Flower Pollination Algorithm overview

The FPA is inspired by the pollination process of flowering plants. Pollination is the transfer of pollen grain from the anther of a flowering plant to the stigma of another flowering plant. Flower pollination aims to maximize the number of reproduced plants, and also increase the number of fittest plants [223]. Some flower pollinators include: insects, honeybees, birds, water, wind and bats. Some of these pollinators (such as honeybees) have a tendency of pollinating only flowers of specific species, and ignoring other accessible flower species. This is referred to as flower constancy [229]. There are two forms of pollination: biotic and abiotic pollination [229]. In biotic pollination, pollinators are responsible for the transfer of pollen grains [223]. However, in abiotic pollination, the transfer of pollens does not require pollinators; wind and water serve as pollinators [223]. Pollinators, such as birds and bats, can transfer pollen between flowers that are far away from each other. They are referred to as global pollinators, because, they can fly over long distances [229]. Global pollination guarantees pollination and reproduction of flowers that are typically fit in the population [223]. There are two types of pollination: cross-pollination and self-pollination [223]. Cross pollination involves transfer of pollen grains from the anther of a flower to the stigma of another flower belonging to a different plant. However, self-pollination is the transfer of pollen grain from the anther of a flower to the stigma of the same flower [223]. Based on these pollination attributes, Yang [223] formulated FPA on four rules, as follows:

i. Processes involved in biotic and cross-pollination is taken as global pollination process, with global pollinators performing levy flight.

ii. Abiotic and self-pollination are taken as local pollination.

iii. Flower constancy is also called reproduction probability. It is proportional to the similarity between two flowers that are involved.

iv. A switch probability p ∈ [0, 1] is used to control global and local pollination. In the pollination process, local pollination can be assigned a significant fraction of p, due to the closeness of some factors, such as wind.

Rule 1 and flower constancy are represented by equation (3.38).

$$x_i^{t+1} = x_i^t + L(x_i^t - g_*), \tag{3.38}$$

where X_i^t refers to vector x_i at different iteration t, and g_* refers to the current best solution in iteration t. Also, L refers to Levy flight, which can be drawn from a levy distribution given in equation (3.39).

$$L \sim \frac{\lambda\Gamma(\lambda)\sin\left(\pi\lambda/_2\right)}{\pi}\frac{1}{s^{1+\lambda}}, (s \gg s_0 > 0), \tag{3.39}$$

$\Gamma(\lambda)$ is a standard gamma function, valid for huge steps, $s > 0$.

Furthermore, rule 2 and flower constancy are represented by equation (3.40).

$$x_i^{t+1} = x_i^t + \in (x_j^t - x_k^t), \tag{3.40}$$

where x_j^t refers to pollen j at iteration t, x_k^t refers to pollen k at iteration t. They refer to pollen grains from different flowers. ∈ is a constant, drawn from the range $[0, 1]$. FPA was designed to handle continuous problem, however, in this research, the rounding-off approach, shown in equation (3.43) is used to convert each flower position to a binary value.

$$S(V_i^t) = \frac{1}{1 + e^{-V_i^t}}, \tag{3.41}$$

Therefore, each position is updated by equation (3.42):

$$X_i^t = \begin{cases} 1 \; if \; rand() \leq S(V_i^t), \\ 0 \quad otherwise, \end{cases}, \tag{3.42}$$

where $rand()$ is a random number selected from the range, $[0, 1]$.

$$\mathrm{X_i^t} = \begin{cases} 1 \text{ if } \mathrm{V_i^t} > 0.5, \\ 0 \quad \text{otherwise}, \end{cases} \tag{3.43}$$

where x_i^t and v_i^t refers to different flower position and velocity at different iterations. t refers to dimension.

3.4.3.2 Flower Pollination Instance Selection Algorithm

FPISA is inspired by the pollinating process of flowering plants. Each flower (or solution) consists of N number of instances, where N is user-defined. FPISA begins by initializing each pollen solution (line 9) and also defining a probability that controls the switch between global and local pollination (line 11). Moreover, all the initialized solutions are evaluated and the best solution is retained (line 15). Furthermore, FPISA continues searching the solution space by performing global or local pollination. Local pollen solutions are generated (using equation (3.40)), if a user-defined probability switch is less than a randomly-generated number (lines 25 to 29). Otherwise, global pollen solutions are generated using equation (3.38). Furthermore, the new solutions are evaluated and the global best solution is updated if a better solution is found. This process is repeated until a user-defined maximum is reached. The algorithm is also terminated if it converges on a solution (lines 34 to 36). After termination, the solution selected by the best flower is used to build SVM model (line 45). Prior to training, if the solution size is less than a user-defined threshold, more instances are selected from the training subset and added to the solution space. This is to ensure that the total number of training instances is not less than the minimum pre-defined value (lines 40 to 42). Pseudocode for FPISA is shown in Algorithm 3.5. For FPISA, the rounding function is used to convert continuous values to binary values.

3.4.4 Social spider-inspired technique

The SSA is a recent NI-based swarm intelligence algorithm proposed by James and Victor [224]. In this thesis, a social spider-based instance selection technique (called SSISA) was designed for improving SVM predictive accuracy and training speed. The section presents an overview of SSA, followed by a description of the proposed social spider-based technique.

3.4.4.1 Social Spider Algorithm overview

The majority of spider species do not relate with each other – they are solitary [224, 230]. Unlike solitary spiders, some spider species exhibit social behaviour [224]. These species reside in groups and relate with each other within the same group [224]. SSA is inspired by the foraging behaviour of social spider species [224].

Spiders are located worldwide [224]. They utilize different methods to scout for food [224]. Spiders are hypersensitive to vibrations, and most of them identify prey by detecting vibrations on their web [224]. Typically, spiders capture prey by analyzing propagated vibrations, and by attacking in the direction of their prey (or source of vibration), if vibration is within a defined frequency range [224, 231]. Moreover, social spiders can differentiate between vibrations stimulated by prey and vibrations stimulated by fellow spiders [224, 232]. One of the reasons animals reside with each other is to increase their chance of capturing prey, and to reduce the cost of energy expended during foraging [233]. There are two types of social foraging models, namely: information-sharing model [234] and producer-scrounger model [235]. In the information-sharing model, food search is performed by foragers, independently, but observes the behaviour of other colony members, to ensure that non-group members do not capture prey that was discovered by fellow colony members [236]. In the producer-scrounger model, foragers in the population are grouped into leaders and followers. The searching method of SSA was formulated based on the information-sharing model, since social spiders have no leader [224]. Also, the problem search space was formulated as a spider web with more than three dimensions. Each web position is a potential solution to the problem solved. All solutions to the problem have their individual position on the spider web [224]. Moreover, spiders are located on individual web positions, and the fitness of each spider is defined by a fitness function [224]. The fitness of each spider represents the probability of obtaining a food source at the spider position [224]. In addition, spiders are free to move around their webs, however, they cannot move out of the web.

Positions out of the web are infeasible solutions [224]. In SSA, each artificial spider holds the following information [224]:

i. Spider position on the web
ii. Current fitness value of spider
iii. Target spider vibration in previous iteration
iv. Number of iterations, after spider vibration was last changed
v. Spider movement in a previous iteration
vi. Dimension mask used by a spider to control movement in a previous iteration

Vibrations are generated at new positions. Vibration intensity is proportional to current spider fitness. In SSA, James and Victor [237] defined vibration using two properties: source position and source intensity of the vibration. The source position is defined by the problem search space, and the source intensity is selected from the range [0, +∞). Vibration intensity is defined mathematically by equation (3.44) [224].

$$I(P_s, P_s, t) = log\left(\frac{1}{f(P_s) - C} + 1\right), \tag{3.44}$$

where P_s, refers to spider position at time t, $I(P_s, P_s, t)$ refers to intensity of vibration generated by spider at source position, $f(P_s)$ refers to fitness of spider at time t, and C is a constant value, where all $f(P_s) > C$. Equation (3.44) considers the following [224]:

i. All vibration intensities are positive
ii. Web positions with high fitness value have higher vibration intensities compared to positions with worse fitness value.
iii. Vibration intensity will not increase comprehensively, if a solution reaches the global optimum. An excessive increase can lead to the malfunctioning of the vibration attenuation scheme.

The distance between two spiders is defined in equation (3.45).

$$D(P_a, P_b) = \|P_a - P_b\|_1, \tag{3.45}$$

where $D(P_a, P_b)$ refers to distance between spider a and spider b. Vibration reduction over a distance is given by equation (3.46).

$$I(P_a, P_b, t) = I(P_a, P_b, t) * exp\left(-\frac{D(P_a, P_b)}{\sigma * r_a}\right), \tag{3.46}$$

where σ refers to standard deviation, and r_a controls the vibration reduction rate over a distance. It is drawn from the range (0, ∞). In SSA, each spider is composed of a dimension mask, of length D, where each bit contains 0 or 1. The dimension mask is used to guide the movement of each spider. Each bit in the dimension mask contains 0 or 1, and they are generated independently using equation (3.47).

$$P_{s,i}^{fo} = \begin{cases} P_{s,i}^{tar} & m_{s,i} = 0 \\ P_{s,i}^{r} & m_{s,i} = 1 \end{cases}, \tag{3.47}$$

Algorithm 3.5: Flower Pollination Instance Selection Algorithm

```
Input: $NF, NI, MaxG, N, NRuns, Min, D, FT$
Output: ACA
1 Start SVM
2 for i = 1 to NRuns
3   for j = 1 to $NF$
4       $TrainingDataset \leftarrow 9/10\ of\ D$ataset /*Get the training dataset for the current fold */
5       $TestDataset \leftarrow 1/10\ of\ D$ataset /*Get the test dataset for the current fold*/
6       TrainingSubset ← RandomSelect(TrainingDataset) /*randomly select training subset from TrainingDataset*/
7       FPISA(TrainingSubset) /*Start instance selection*/
8           Initialize Parameters /*initialize all the algorithm parameters*/
9           Define $G(x)$ for flowers /*define fitness function for both filter and wrapper-based FPISA*/
10          Define $PS, PS \in [0, 1]$ /*define probability switch for flowers*/
11          for $a$ = 1 to $N$
12              Initialize solution for $spider_a$ /*initialize the solution for all flowers in the solution space*/
13          end for
14          Evaluate $G(x)$, and select $CB$ /*Evaluate the objective function for all solutions & select the current best*/
15          $GB \leftarrow CB$ /*Retain the current best solution*/
16          while ($p < MaxG$) /*Start searching for new pollen solutions */
17              for $k$ = 1 to $N$
18                  $R$ ← RandomNumber() /*generate random number R, where $R \in [0,1]$*/
19                  if $R > PS$ /*if this is true, perform global pollination*/
20                      for $l = 1\ to\ Dim$ /*define levy flight factor for global pollinators*/
21                          Randomly generate $LF$ vector for each dimension
22                      end $l$
23                      Perform global pollination using $x_i^{t+1} = x_i^t + LF(x_i^t - g_*)$,
24                  else /*perform local pollination*/
25                      $R$ ← RandomNumber() /*generate random number R, where $R \in [0,1]$*/
26                      Randomly select two solutions, $x_j^t, x_k^t$ from population
27                      Perform local pollination using $x_i^{t+1} = x_i^t + \in (x_j^t - x_k^t)$
28                  end if
29                  Convert solutions to binary using equation (3.43)
30              end k
31              Evaluate $G(x)$ /*evaluate the fitness value for all the new solutions*/
32              $GB \leftarrow CB$ /*Update the global best with the current best solution*/
33              if $GB > FT$
34                  end while /*Stop algorithm if global best is greater than a pre-defined fitness threshold*/
35              end if
36              $p{+}{+}$
37          end while
38          $NS \leftarrow GB$
39          if $NS < Min$ /*Add more instances if the number of instances is less than a user-defined threshold*/
40              AddInstances(GB) /*Add (Min-NS) instances to the instances selected by the global best*/
41          end if
42          Output $GB$ /*Output the global best solution*/
43      end FPISA
44      TrainSVM(GB) /*Train SVM on the solution selected by FPISA*/
45      $CA{+}{=}\ evaluateModel(TestSubset)$
46  end j
47  $ACA{+}{=}\ CA\ /\ NF$ /*Add the predictive accuracy produced by the current fold*/
48 end i
49 $ACA \leftarrow CA\ /\ NF$
50 Output ACA / NRuns
51 end FPISA
```

where $P_{s,i}^{fo}$ refers to i-th dimension of spider s at position i, r is a random value from the range $[1, |pop|]$. pop refers to spider population and $|pop|$ refers to the number of vibrations generated by spiders in the population. Also, $m_{s,i}$ represent i-th dimension of dimension mask m of spider s, $P_{s,i}^{tar}$ is the i-th source position of the target vibration of spider s. After the dimension mask for each spider has been generated, each spider performs a random walk, using equation (3.48).

$$P_s(t+1) = P_s + \left(P_s - P_s(t-1)\right) * r + \left(P_s^{fo} - P_s\right) \odot R, \tag{3.48}$$

where $\odot$ represents element-wise multiplication, and R is a random number, uniformly generated in the range [0, 1]. During movement, there is a possibility of a spider moving out of the web, which will violate the constraints of the optimization problem at hand. Hence, in SSA, each spider position is controlled by equation (3.49), which handles the boundary constraints.

$$P_{s,i}(t+1) = \begin{cases} \left(\overline{x_i} - P_{s,i}\right) * r & if\ P_{s,i}(t+1) > \overline{x_i} \\ \left(P_{s,i} - \underline{x_i}\right) * r & if\ P_{s,i}(t+1) < \underline{x_i} \end{cases} \tag{3.49}$$

where $\overline{x_i}$the upper bound of the problem search space in dimension i, and $\underline{x_i}$ is the lower bound of the problem search space of the i-th dimension. r represent a random number generated between 0 and 1. In this paper, standard SSA, proposed by James and Victor [224], was used. It was designed to handle problems in continuous space, however, since instance selection is a binary problem, the sigmoid function (defined in equation (3.41)) is used to convert each spider position to a binary value. In addition, each spider position is updated by equation (3.42).

3.4.4.2 Social Spider Instance Selection Algorithm

SSISA is inspired by the foraging behaviour of social spiders. SSISA begins by initializing all parameters and generating an initial solution of N spiders, where each spider consists of d instances (line 10 and 12). The vibration intensity for each spider is also initialized (line 13). In addition, the fitness score for each spider is calculated and the spider with the best fitness value is stored (line 16). Moreover, new solutions are generated by moving each spider to different positions on the web (lines 17 to 41). Each spider movement causes a vibration, as calculated in equation (3.44). Typically, spiders capture prey based on propagated vibrations, and they attack the prey direction (or source of vibration) if the vibration is within a defined frequency range [224, 231]. In SSISA, if the vibration generated by the current solution is greater than a pre-defined target vibration, then the target vibration is updated with the best vibration. Furthermore, a random number is generated and the instance mask is updated if the random number is greater than a pre-defined threshold (line 26 to 28). Further, the position of each spider is generated (line 30) and the fitness value for the newly generated solutions are computed and the current best solution is compared to the global best solution (line 34). If it is better, it is retained, otherwise it is discarded. The process is performed repeatedly until a stop criteria is reached. Furthermore, after the algorithm terminates, the instances selected by the best spider solution is used to train SVM and the predictive accuracy is outputted. Before training, the number of instances selected by the best solution are checked to ensure that they are not less than the minimum threshold (lines 42 to 44). Pseudocode for SSISA is shown in Algorithm 3.6. SSA is designed to work in a continuous space, hence SSISA uses sigmoid function to convert continuous value to binary.

3.4.5 Firefly-inspired technique

This thesis proposes a firefly-based instance selection solution for improving the training speed and predictive accuracy of SVM. The section begins with an introduction to the standard FFA, followed by a description of the proposed firefly-based technique.

3.4.5.1 Firefly Algorithm overview

FFA is inspired by a distinctive attribute of fireflies – their flashing light. About 2,000 firefly species exist, and most of these species produce short flashes at consistent time intervals [238]. Flashlights are produced to entice mating partners and prey and also to warn possible predators away from attacks. FFA is suitable for handling challenging NP-hard and optimization problems [239]. The light intensity of the firefly flashlight decreases with every increase in distance, that is, light intensity is inversely proportional to the distance squared, as shown in equation (3.50).

$$I \propto {}^{1}/_{r^2} \tag{3.50}$$

Also, the flashlight is absorbed into the atmosphere as the distance increases, which consequently leads to a decrease in the light intensity. As pointed out by Yang [238], the flashlight can be formulated in a manner that will be proportional to the fitness function. Some variants of FFA exist in the body of literature, however, this research utilized the original version of the firefly proposed by Yang [238]. FFA was designed using three rules:

i. All firefly species are of the same sex.

ii. Fireflies' attractiveness is proportional to the intensity of light they produce. This implies that fireflies with high light intensity will attract fireflies with lower light intensity.
iii. Fireflies' light intensity is determined by the landscape of the fitness function to be improved.

Light intensity and attractiveness are two vital issues that need to be clearly defined when utilizing FFA. Generally, for maximization problems, firefly light intensity (I) produced at a given location (y), is directly proportional to the fitness value of the objective function. That is, $I(y) \propto F(y)$. Light intensity produced by fireflies changes with changes in firefly distance. It also changes with respect to the intensity of light absorbed by the atmosphere, as shown in equation (3.51):

$$I(r) = I_0 e^{-\gamma r^2} \tag{3.51}$$

where I_0 is the initial light intensity when r=0, γ is a constant representing the light absorption coefficient, and r represents the distance. In equation (3.51), Yang [238] notes that the singularity at r = 0 is avoided in the expression $1/r^2$, by merging the effect of the inverse square law and absorption. Also, the singularity is avoided by approximating them in Gaussian form as shown in equation 3.51. Also, firefly attractiveness (β) is proportional to their light intensities as shown in equation (3.52):

$$\beta = \beta_0 e^{-\gamma r^2} \tag{3.52}$$

where β_0 is the attractiveness at r = 0.

The distance between two fireflies (x_i and x_j) is calculated by the Euclidian distance, as shown in equation (3.53):

$$r_{ij} = \| x_i - x_j \| = \sqrt{\sum_{k=1}^{d} \left(x_{i,k} - x_{j,k}\right)^2} \tag{3.53}$$

where d is the problem dimensionality. A Firefly moves from one point (point i) to another (point j) according to equation (3.54):

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}\left(x_j - x_i\right) + \alpha \epsilon_i \tag{3.54}$$

$\alpha \in [0,1], \gamma \in [0, \infty)$. ϵ_i are two random numbers taken from a Gaussian distribution, ϵ_i can be substituted by $rand - {}^{1}/_{2}$ where $rand \in [0,1]$. The second term in equation (3.54) shows the movement of a firefly as a result of their attractiveness to fireflies with brighter light

Algorithm 3.6: Social Spider Instance Selection Algorithm

```
Input: $NF, NI, MaxG, N,$ NRuns, $D, Pm, FT$
Output: $ACA$
1 Start SVM
2 for i = 1 to NRuns
3     for j = 1 to $NF$
4         $TrainingDataset$ ← 9/10 of dataset
5         $TestDataset$ ← 1/10 of dataset
6         TrainingSubset ← RandomSelect(TrainingDataset) /*randomly select training subset from TrainingDataset*/
7         SSISA(TrainingSubset) /*start instance selection*/
8                 Define $G(x)$ for spiders /*pass the selected training subset to FPISA for processing*/
9                 Initialize Parameters /*initialize all the algorithm parameters*/
10                for $a$ = 1 to $N$
11                        Initialize solution for $spider_a$ /*initialize the solution for all spiders in the solution space*/
12                        Initialize vibration ($TV$) for $spider_a$ /*generate initial vibration for each spider*/
13                end for
14                Evaluate$G(x)$, /*evaluate the fitness of the initial solution*/
15                $GB \leftarrow CB$ /*if current best is greater than global best, update the global best solution*/
16                while ($p < MaxG$) /*start search for more solution*/
17                        for $k$ = 1 to $N$
18                                Calculate $VI$ generated by all spiders and select $GBV$ /*select the best bat vibration*/
19                                if $GBV > TV_k$ /*if the best vibration is greater than a user defined target vibration*/
20                                        $TV_k = G$BV /*update the target vibration*/
21                                end if
22                                Update $Tot_k$ /*keep track of frequency of vibration change*/
23                                for $a$ = 1 to $D$
24                                        Generate Random Number, $R$ where $R \in [0, 1)$
25                                        If $R1 > P$
26                                                Update $dimension_a$ for $spider_k$ /*update dimension mask */
27                                        end if
28                                end $a$
29                                Generate new position for $spider_k$
30                                Do Random Walk, and handle violated boundary constraints
31                                Convert $spider_k$ to binary using sigmoid
32                        end $k$
33                        Evaluate $G$(x) for new $solution_a$ and generate vibration for $spider_a$
34                        Convert $spider_a$ to binary using sigmoid function
35                        Evaluate $G(x)$ for new solutions, and update $GB$ accordingly
36                        if $GB > FT$
37                                end while /*Stop algorithm if global best is greater than a pre-defined fitness threshold*/
38                        end if
39                        $p{+}{+}$
40                end while
41                if $NS < Min$
42                        update $GB$ by adding (Min - NS) instances to GB
43                end if
44                Output $GB$
45        end SSISA
46        Train SVM model on instances selected by $GB$
47        $CA{+}{=} evaluateModel(TestSubset)$
48    end j
49    $ACA{+}{=} CA / NF$ /*Add the predictive accuracy produced by the current fold*/
50 end i
51 $ACA \leftarrow CA / NF$
52 Output $ACA$ / NRuns /*Output the overall average*/
53 end SVM
```

intensity. When $\beta_0 = 0$, a firefly will move randomly. In this study, the rounding-off approach as defined in equation (3.43) is used to convert each firefly position from continuous values to binary values.

3.4.5.2 Firefly Instance Selection Algorithm

Similar to FFA, FFISA is inspired by the flashing behaviour of fireflies. Given a set of training instances, FFISA is used to select the best subset of relevant instances for training. Each firefly consists of a binary array of N instances (called instance mask), where 1 indicates that an instance is selected, and 0 indicates otherwise. As shown in Algorithm 3.7, FFISA begins by initializing the firefly parameters and randomly initializing each firefly position to 0 and 1 (lines 9 and 10). Furthermore, the objective function for each firefly is evaluated and the global best solution is retained (lines 11 and 12). The global best solution is the solution with the brightest light intensity. Furthermore, FFISA search for new solutions by moving each firefly to new positions within the search domain, based on their attractiveness level (lines 13 to 19). Fireflies with low light intensities are moved from their positions to fireflies with higher light intensities using equation (3.54). After fireflies have been moved to different positions, the fitness value for each new solution is evaluated and the global best solution is updated if a better solution is found (line 22). This process is repeated until a pre-defined number of generations is reached (line 26) or until the algorithm converges to a desired solution (line 24). Furthermore, after termination, FFISA selects the firefly with the highest attractiveness value (i.e. the global best) and extracts the selected instances for training. If the number of selected instances is less than a user-defined threshold, then more instances are randomly added to the solution. This is to ensure that the data size that is used to train SVM is not less than a minimum value. Finally, the selected instances are used to train SVM. FFA was designed to work in continuous space, hence, FFISA uses a rounding-off function to convert continuous values to binary values.

Algorithm 3.7: Firefly Instance Selection Algorithm

```
Input:   NR, NRuns
Output: ACA
1 Start SVM
2 for i = 1 to NRuns
3    for $j$ = 1 to $NF$
4       $TrainingDataset \leftarrow 9/10\ of\ D$ /*Select Training dataset – 90% of the entire dataset*/
5       $TestDataset \leftarrow 1/10\ of\ D$ /*Select Test dataset – 10% of the entire dataset*/
6       TrainingSubset ← RandomSelect(TrainingDataset) /*randomly select training subset from TrainingDataset*/
7       FFISA(TrainingSubset) /*pass the selected training subset to FPISA for processing*/
8               Initialize firefly parameters: $NF, NG, \beta o, \alpha, and\ \gamma$ /*initialize all the firefly parameters*/
9               Generate initial populations of fireflies $x_i$ (i = 1,2,…NFF)
10              Evaluate $G(x)$ to obtain $L_i$ for each firefly /*Evaluate initial solutions, to get firefly light intensities*/
11              Rank firefly and save $GB$ /*rank the current solutions and retain the current best*/
12              while (n < MaxGen)
13                 for $p$ = 1 to $NFF$ /*start searching for better solutions*/
14                    for $q$ = 1 to $NFF$
15                       if ($l_p < l_q$) /*move fireflies based on their individual light intensities*/
16                          Move $firefly_p$ towards $firefly_q$ using equation (3.54)
17                       end if
18                    end $q$
19                 end p
20                 $n$++
21                 Evaluate $G(x)$ /*evaluate the new solutions*/
22                 Rank firefly and save $GB$ /*rank the current solutions and retain the current best*/
23                 if $GB > FT$
24                    end while /*Stop algorithm if global best is greater than a pre-defined fitness threshold*/
25                 end if
26              end while
27              $NS \leftarrow GB$ /*Assign the instances selected by the best solution to NS*/
28              if $NS < Min$ /*if the selected instances is less than a predefined minimum, then add more instances*/
29                 update GB by adding ($Min - NS$) instances to $GB$
30              end if
31              Output $GB$
32      end FFISA
33      Train SVM on instances selected by GB
34      $CA += evaluateModel(TestSubset)$
35   end j
36   $ACA += CA / NF$ /*Add the predictive accuracy produced by the current fold*/
37 end i
38 $ACA \leftarrow CA / NRuns$ /*Compute the overall predictive accuracy*/
39 Output ACA
40 end SVM
```

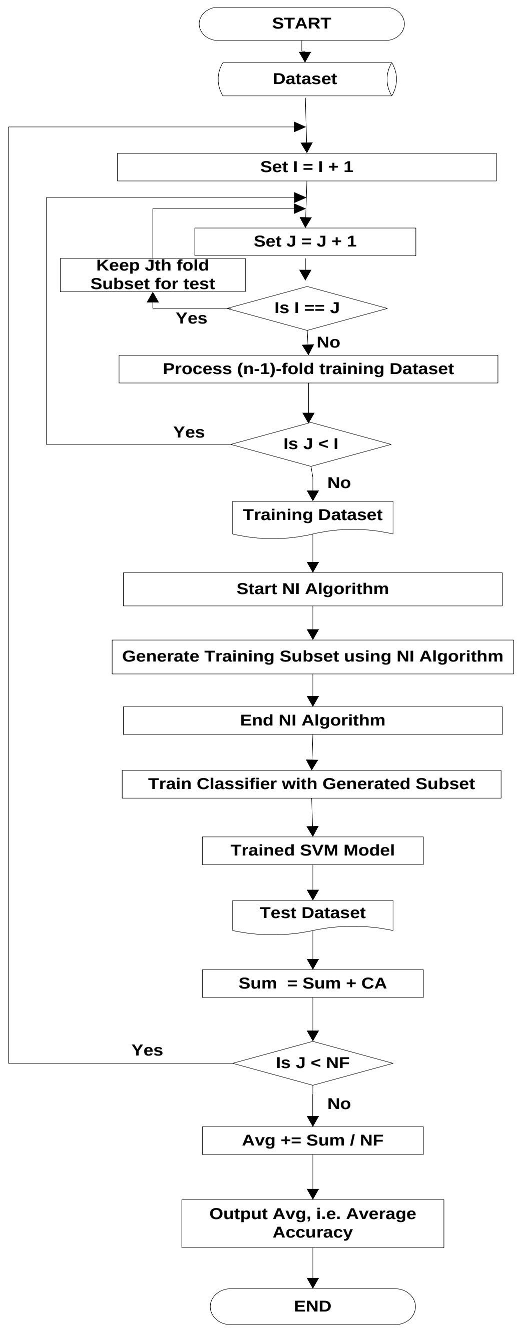

Figure 3.5: Flowchart for the proposed NI-based algorithms

3.5 Fitness function

This thesis proposes intelligent filter-based and wrapper-based instance-selection techniques for improving SVM performance. The filter-based techniques are designed to improve SVM classification speed and the wrapper-based techniques are designed to improve SVM predictive accuracy. The main difference between both techniques is in their selection criterion. The filter-based techniques use equation (3.55) to evaluate the fitness for each candidate solution, while the wrapper-based techniques use the SVM classifier to evaluate the fitness for each candidate solution. More details on the filter-based and wrapper-based fitness functions are provided in Sections 3.5.1 and 3.5.2, respectively.

3.5.1 Fitness function for filter-based techniques

This section present a description of the novel selection function proposed in this thesis. As shown in equation 3.55, the selection 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 computing the total number of instances in each agent (α). Further, the evaluation continues by calculating 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. The number of boundary instances selected by the algorithm is then calculated and used for fitness value evaluation. In this research, a clustering-based boundary detection algorithm, proposed by Chen *et al.* [1], is used for boundary instance selection. Finally, α, β and γ are used to calculate the fitness value, as shown in equation (3.55).

$$fitness_i = \left(\left(100 * \frac{\alpha-\beta}{\alpha}\right) + \left(\frac{\gamma}{\beta} * 100\right)\right) \Big/ 2 \tag{3.55}$$

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

3.5.2 Fitness function for wrapper-based techniques

The fitness function utilized by the wrapper-based instance selection techniques is shown in equation (3.56). The fitness function evaluation starts by computing the predictive accuracy (α) of the candidate solution (i.e. reduced subset) constructed by each agent. That is, for each candidate solution, a classification model is constructed by training the generated solution on a classifier. Afterwards, the model is evaluated by validating it on a new dataset (test dataset), and the resultant classification accuracy is used as the fitness value for the candidate. The candidate (or subset) with the best fitness value is the subset with the highest classification accuracy. Finally, after a user-defined threshold has been reached, the best subset is selected and used to build the final classifier. Take note that the test set is completely different from the training set.

$$fitness_i = \alpha_i \tag{3.56}$$

where α_i is the classification accuracy produced after validating each candidate on the test set

3.6 Features used for classification

This section provide details on the features extracted from the spam email and phishing email datasets used for evaluation. The credit card fraud dataset used for evaluation was already processed by dataset providers [240], hence feature extraction was not necessary. Moreover, dataset providers did not provide details on the extracted credit card fraud features, due to confidentiality issues.

3.6.1 Spam email features

Prior to classification, some set of spam features were 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 [34], 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.

3.6.1.1 Word-Based Features

For this feature, different words are extracted from all emails in the dataset, using the extraction technique proposed by Paul Graham [241]. Moreover, the spam score for each word is calculated, and the 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.

3.6.1.2 Term Frequency + Inverse Sentence Frequency

This feature is a combination of TF and inverse sentence frequency (ISF). For each email, TF for each word is calculated using equation (3.57), and ISF for each sentence is calculated using equation (3.58). Finally, as shown in equation (3.59), the sum of the product of TF and ISF is calculated and used as a feature. In this study, this feature was converted 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 [242].

$$TF_t = 1 + log(frequency)\,, if\ frequency_t > 0, 0\ otherwise \tag{3.57}$$

$$ISF_t = log\frac{N}{SF_t}, \tag{3.58}$$

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

$$\sum_t TF_t \times ISF_t \tag{3.59}$$

3.6.1.3 Complex Words

Words with more than two syllables are called complex words. For this feature, 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 [242].

3.6.1.4 Simple Words

The term of simple words refers to words with one or two syllables. A Boolean value of 0 is recorded if an email contains less than fifty simple words, and 1 is recorded if an email contains more than fifty simple words. This feature is similar to the feature used by Shams and Mercer [242].

3.6.1.5 Spam Words

Some list of spam words, provided by Sham and Mercer [242], is extracted and used as features. A Boolean value of 1 is recorded if an email contains more than one spam word, otherwise a value of 0 is recorded.

3.6.1.6 Total HTML Tags

HTML tags are keywords that define how web browsers format and display content [243], such as text and images. HTML tags are extracted from each email and a Boolean value of 1 is recorded if an email contains more than one HTML tag otherwise a value of 0 is recorded. This feature was also used by the authors in [242].

3.6.1.7 Document Length

Document length refers to the number of sentences in an email document. A Boolean value of 1 is recorded if an email contains more than one sentence, otherwise a value of 0 is recorded. This feature was proposed by Shams and Mercer [242].

3.6.1.8 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 are assigned the value of one, and emails containing one or no non-anchor tag are assigned the value of zero. This feature was also used by Shams and Mercer [242].

3.6.1.9 Stop Words

Stops words are words frequently used in a specific language. Some list of stop words, provided by Shams and Mercer [242], is extracted from each email and a Boolean value is recorded. Emails with stop words greater than one hundred in number, are assigned the value of one, and emails containing less than one hundred stop words are assigned the value of zero. This feature was proposed by Shams and Mercer [242].

3.6.1.10 Presence of ‘Link’, ‘Click Here’ in URL Text of a Link

Most spam or phishing email typically require users to click on a Web link, which re-directs them to a spam or phishing Website. 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. A similar feature is used by the authors in [244].

3.6.1.11 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 by the authors in [244] and [245].

3.6.1.12 Sum of Distinct Domain

Domain names are used to detect Web pages. For this feature, domain names are extracted from each email and the total number of domain names is recorded and used as a continuous feature. Domain names that appear more than once are counted only once. This feature was also used by the authors in [89] and [244].

3.6.1.13 SpamAssassin Feature

SpamAssassin is a reliable spam email filter and is currently used by some organizations. In this research, 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 features were used by Akinyelu *et al.* [244] and Fette *et al.* [89].

3.6.1.14 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.* [89], note 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 [244] and [89].

3.6.1.15 Total Email Links

Zhang and Y. Yuan [246] note that emails containing many URLs are likely spam or phishing emails. Hence, email links are extracted from each email and the total number of Web links are recorded and used as a continuous feature. This feature was also used by the authors in [244] and [246].

3.6.2 Phishing e-mail features

The "Bag-of-words" approach used in spam filtering is not similar to the approach used for phishing email classification [70]. This is because techniques used for spamming (such as typographic errors) do not frequently appear in phishing emails. Hence, phishing-specific features are most adequate for the filtering of phishing emails. This section presents the details of the phishing features used in this study. Six of the features used for phishing emails are similar to the spam email features already presented in Sections 3.6.1.10 - 3.6.1.15. Hence, they are not presented in this section.

3.6.2.1 IP-Based URLs

Generally, Website URLs of legitimate organizations or companies typically contain the name of the company or organization. For example, "www.google.com" informs users that the URL belongs to Google International. However, phishing URLs typically contain some string of numbers, called IP address (for example, "http://145.21.455.12/login.yahoo.com"). Phishers use these numbers to hide Website names from users. A Boolean value is recorded for each email, based on whether the email contains IP-based URLs. This feature was also used by the authors in [6] and [244].

3.6.2.2 Disparities between 'href' attribute and LINK text

URLs are used to access Web pages on Internet. URLs can be coded in an email using the HTML anchor tags. For example, "<a href="www.google.com">Google<a/>". As shown in the example, the "href" attribute is used to define the actual address of the Website ("www.google.com"). A user is expected to click on the link text ("Google" in the above example), before the browser is directed to the Website. The URL defined by the "href" attribute and the string specified in the link text should be the same. Hence, for this feature, a binary value is recorded based on whether there is a disparity between the "href" attribute and the link text in an email. This feature was used by the authors in [244] and [6].

3.6.2.3 Number of Domain Name Dots

URLs for legitimate organizations or companies should not contain more than three dots [245] (for example, www.yahoo.com contains two dots). Hence, for this feature, a binary value of 1 is recorded if an email contains more than 3 dots, otherwise a value of 0 is recorded. This feature was also used by the authors in [244].

3.6.2.4 Presence of JavaScript

Phishers typically use the JavaScript programming language to mask information from end users. JavaScript can be coded in the body section of an email (using the script (<script>) tag) or in a URL (using the anchor (<a>) tag). Therefore, a binary value is recorded based on the presence or absence of the word “javascript” in either the body section of an email or in a URL contained in the email. This feature was also used by the authors in [89] and [244].

3.6.2.5 Word-Based Features

The following groups of words are extracted, counted, normalized and used as features. These groups of words often appear in phishing emails. Prior to extraction, the words were grouped into batches of six, and each group were used as a single feature. Hence, a total of six word-based features is extracted from each email. This feature is similar to one of the features used in [247]. The groups of words include:

i. Social Security, SSN
ii. Customer, User, Client
iii. Confirm, Update
iv. Account, Verify
v. Username, Password, Login
vi. Hold, Restrict, Suspend

3.7 Chapter summary

This thesis proposed seven filter-based and five wrapper-based intelligent instance selection techniques for improving SVM speed and predictive accuracy. Two of the proposed filter-based techniques are boundary detection algorithms that are inspired by edge detection techniques in image processing and edge selection techniques in ACO, respectively. The remaining five filter-based and wrapper-based techniques are based on the following NI algorithms: CSA, FPA, SSA, FFA and BA. The primary difference between the filter- and wrapper-based techniques is in their method of selection. The fitness function of the filter-based techniques is designed with the primary objective of improving SVM training speed, while the fitness function of the wrapper-based techniques is designed with the primary objective of improving SVM predictive accuracy. This section presents a detailed description of the proposed wrapper-based and filter-based techniques. Different sets of experiments are performed to validate the efficiency of the proposed techniques. Moreover, some set of spam and phishing email features (also presented in this section) was extracted and used to train and build SVM classification models. The experimental results produced by the proposed filter-based techniques reveal excellent improvement in SVM training speed. Moreover, the experimental results produced by the wrapper-based techniques show improvement in SVM predictive accuracy. Detailed information on the results are presented and discussed in Chapter 4.

Chapter 4
Experimental Setup, Dataset and Results

Two sets of experiments are performed to evaluate the performance of the proposed techniques. The first set of experiments was performed to evaluate the performance of the proposed filter-based techniques and the second set of experiments was performed to evaluate the performance of the wrapper-based techniques. This section provides information on the results obtained from the experiments. This section also provides information on the experimental setup and the datasets used for the experiments. In addition, this section provides details on the methods used to evaluate the performance of the proposed technique.

4.1 Experimental setup

All experiments are performed on a Core i7 computer, operating on Windows 7, 64 bits and 3.10GHz with 8GB of RAM. Moreover, for all the experiments, the RBF kernel is used. The RBF kernel requires the selection and tuning of two parameters: $C \text{ and } \gamma.$ As recommended by Hsu [27], different exponential growing sequences of $C \text{ and } \gamma$ pairs are tested, and the best $C \text{ and } \gamma$ pair is selected and used for training. Table 4.1 shows the sequence of RBF parameters used for all experiments in this research. Also, Tables 4.2 and 4.3 report the values for other parameters.

Prior to training, some set of features (described in Section 3.6) were programmatically extracted using C# programming language. Sixteen features was extracted from a dataset consisting of phishing and legitimate emails (dataset A), and a set of fifteen features was extracted from a dataset consisting of spam and legitimate emails (dataset B). Feature extraction was not necessary for the credit card fraud dataset (dataset C) and spambase dataset (dataset D), because the two datasets were already processed by their providers. Furthermore, all the extracted features were processed and converted to the input format required by libSVM [34] – the SVM library used in this research. Specifically, Matthew Johnson DotNet implementation [35] of libSVM is used in this research. All features are scaled down using Gaussian transformation. Scaling ensures that all feature vectors have a mean of zero and a standard deviation of one [75]. Additionally, for datasets A and B, all the extracted features are further reduced using IG. For dataset A, IG for all the sixteen extracted features was calculated, and the best nine features were selected and used for training. Also, for dataset B, IG was calculated for all the extracted features, and the best ten features were used to train SVM. Dataset C contains fifty-seven features, and all of the features were used for training and testing. Also, Dataset D contains 28 features and all the features were used for training and testing. All the experiments were performed using the popular, 10-times, 10-fold cross validation.

Result for different subset sizes and different K values are reported for the proposed boundary detection algorithms (i.e. EDISA and ACOISA). Subset size refers to the number of instances in the training subset processed by EDISA and ACOISA. K refers to the number of nearest neighbours selected by EDISA and ACOISA, for training. Moreover, the results for different subset sizes and different number of particles (NP) are reported for the proposed NI-based techniques (i.e. FFISA, FPISA, SSISA, BISA and CSISA).

4.2 Performance measure

There are four prevailing possibilities in binary classification-related tasks [248], namely: True Positive (TP, illegitimate emails properly classified as legitimate), FP (legitimate emails incorrectly classified as illegitimate), True Negative (TN, legitimate emails properly classified as legitimate) and FN, illegitimate emails incorrectly classified as legitimate. In all experiments performed in this research, the performance measures used are defined in equations (4.1) – (4.7):

$$\text{Average Classification Accuracy} = \frac{CA}{TNR} \tag{4.1}$$

$$\text{Storage Reduction} = \frac{TR}{TT} * 100 \tag{4.2}$$

$$\text{FP Rate} = \frac{TP}{FP + TN} \tag{4.3}$$

$$\text{FN Rate} = \frac{FN}{TP + FN} \tag{4.4}$$

$$\text{Precision (Pr)} = \frac{TP}{TP + FP} \tag{4.5}$$

$$\text{Recall (R)} = \frac{TP}{TP + FN} \tag{4.6}$$

$$\text{F-Measure (FM)} = \frac{2 * Pr * R}{Pr + R} \tag{4.7}$$

where, CA, TNR, TR, TT denote Classification Accuracy, Total Number of Runs, Total number of Retained Instances, Total number of Training instances, respectively. Additionally, total time taken (in seconds) for training and testing is reported.

4.3 Dataset information

The proposed techniques have been validated on 24 datasets. The first dataset (dataset A) contain 3500 ham emails, and 500 spam emails. The second dataset (dataset B) contain 3500 ham emails and 500 phishing email. The ham and spam emails in datasets A and B were obtained from SpamAssassin [31]. The phishing emails were obtained from monkey.org [33]. The phishing emails are no longer available Online. For access to these, interested users should contact the dataset provider, Jose Nazario [33]. The third dataset (dataset C) consist of 1813 spam emails and 2787 ham emails, provided by UCI data repository [249]. The fourth dataset (dataset D) contains 492 credit card fraud and 4508 legitimate credit card transactions, provided by Andrea [240]. The remaining 20 datasets are provided by UCI data repository [249]. Table 4.4 reports a summary of the datasets used in this research.

Table 4.1: SVM parameters used for evaluations

SVM Parameters [27]	C =	2^{-11}	2^{-9}	…………………………	2^{1}	2^{3}	2^{5}
	γ =	2^{-5}	2^{-3}	…………………………	2^{7}	2^{9}	2^{11}

C = regularization constant, γ = Gamma

Table 4.2: Parameters used for experiments

Technique						
FFISA	$\boldsymbol{\alpha}$	$\boldsymbol{\gamma}$	$\boldsymbol{\beta_0}$	$\boldsymbol{N_{ff}}$	$\boldsymbol{N_g}$ **(Filter)**	$\boldsymbol{N_g}$ **(Wrapper)**
	0.2	1	1	20	5	3
FPISA	**Probability Switch**	$\boldsymbol{N_g}$**(Filter)**	$\boldsymbol{N_g}$**(Wrapper)**			
	0.8	5	3			
SSISA	**Attenuation Rate**	**Probability Change**	**Assigning Probability**	$\boldsymbol{N_g}$**(Filter)**	$\boldsymbol{N_g}$**(Wrapper)**	
	1	0.7	0.1	5	3	
CSISA	**Discovery Rate**	**Tolerance**	$\boldsymbol{N_g}$**(Filter)**	$\boldsymbol{N_g}$**(Wrapper)**	**Beta**	
	0.25	$1.0e^{-5}$	5	3	1.5	
BISA	**Loudness**	**Pulse Rate**	$\boldsymbol{N_g}$**(Filter)**	$\boldsymbol{N_g}$**(Wrapper)**	**Minimum Frequency**	**Maximum Frequency**
	0.5	0.5	5	3	0	2

Key: α = alpha, γ = Gamma, β_0 = Beta, N_{ff} = Number of firefly, N_g = Number of generations

Table 4.3: Parameter used for ACOISA

Alpha	Beta	Number of Neighborhood	Evaporation Rate	Total Ant Movement	Decay Coefficient	Initial Heuristic Value	Iteration
1	2	8	0.1	40	0.05	0.01	10

Table 4.4: Dataset used for experiments

Dataset Name	Size	Ham	Spam/Phishing
Dataset A	4000	3500	Spam: 500 (12.5%)
Dataset B	4000	3500	Phishing: 500 (12.5%)
Dataset C	4600	2787	Spam: 1813 (39.4%)
Dataset D	5000	4508	Credit Card: 492 (9.84%)
Abalone	4177	-	-
Balance Scale	625	-	-
Breast Tissue	106	-	-
Bupa	345	-	-
Credit-g	1000	-	-
Cleaveland	303	-	-
Ecoli	336	-	-
Glass	214	-	-
Hungarian	294	-	-
Iris	150	-	-
Liver	345	-	-
Pima Indians	768	-	-
Post Operative	87	-	-
Transfusion	748	-	-
Vertebral-3c	310	-	-
Voting	435	-	-
Waveform	500	-	-
Wine	178	-	-
Yeast	1484	-	-
Zoo	101	-	-

4.4 Results and discussion

This section reports the experimental results produced by the proposed filter and wrapper-based instance selection techniques. The results of standard SVM and two existing filter-based instance selection techniques, CLUS [1] and KNN [8], are discussed in this section. In Section 4.4.1, the results of the proposed filter-based techniques are discussed and compared to the results of the standard SVM and seven existing filter-based instance selection techniques. Also, in Section 4.4.2, the results obtained by the proposed wrapper-based techniques are presented and compared to the standard SVM and a wrapper-based instance selection technique (ADRMiner [250]). Finally, statistical test result are reported in Section 4.4.3. The following performance measures are reported: Average Classification Accuracy (ACA), Global Best (GB) accuracy, False Positive (FP) rates, FN rates, Recall (R), Precision (Pr), F-Measure (FM), Time (T) in seconds and storage reduction.

Tables 4.5 – 4.7 show the experimental results produced by standard SVM, CLUS [1] and KNN [8], for credit card fraud, phishing emails and spam emails. Standard SVM refers to SVM without data reduction. CLUS [1] and KNN [8] are two existing filter-based instance selection techniques adopted in this research for the primary purpose of comparison. As shown in Table 4.5, standard SVM obtained good predictive accuracy for credit card fraud, spam email and phishing email classification. Also, standard SVM produced better predictive accuracy when applied to phishing emails, compared to spam emails and credit card fraud. This is because of the quality of the phishing features used for training. Furthermore, as shown in Table 4.5, SVM training speed decreases, as dataset size, feature size and number of classes increase. SVM performs slower for credit card fraud compared to spam and phishing emails. This is because the credit card fraud dataset used for experiments contains more features and instances, compared to the spam and phishing email datasets.

Tables 4.6 and 4.7 show the results for CLUS [1] and KNN [8], respectively. As shown in the tables, both techniques produced good classification accuracy for credit card fraud detection, phishing and spam email classification. CLUS [1] outperformed KNN [8], in terms of predictive accuracy, however, KNN [8] outperformed CLUS [8], in terms of classification speed. This is because KNN selected fewer instances for training compared to CLUS [8]. As shown in

Table 4.6, CLUS [8] selected over 41% of the training dataset and KNN [8] selected less than 14% of the training datasets.

Table 4.5: Standard SVM results for e-fraud detection

Mail Type	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)
Credit Card Fraud	98.83	99.4	0.29	9.23	90.77	97.07	93.79	2072.99
Phishing Email	99.66	100	0.08	2.2	97.8	99.47	98.52	943.24
Spam Email	96.66	97.5	3.13	4.8	95.2	81.28	87.62	953.94

Table 4.6: CLUS [1] results for e-fraud detection

Mail Type	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage
Credit Card Fraud	98.47	99.4	0.46	11.31	88.69	95.48	91.9	684.06	41.67
Phishing Email	99.53	100	0.23	2.16	97.84	98.47	98.03	337.46	41.67
Spam Email	96.44	100	2.61	10.28	89.72	84.41	85.35	311.98	41.67

Table 4.7: KNN [8] results for e-fraud detection

Mail Type	Subset Size	K	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage
Credit Card Fraud	400	200	91.67	98.6	8.68	5	95	57.73	70.72	297.84	8.89
	400	300	90.21	99	10.37	4.55	95.45	53.92	67.67	230.85	8.89
	500	200	92.50	97.4	7.54	4.41	95.59	58.83	72.26	259.22	11.11
	500	300	92.04	96.8	8.37	4.12	95.88	57.38	71.18	268.75	11.11
Phishing Email	400	200	99.59	100	0.25	1.56	98.44	98.34	98.3	244.15	5.56
	400	300	99.55	100	0.27	1.79	98.24	98.21	98.11	219.51	8.33
	500	200	99.40	100	0.48	1.4	98.6	96.9	97.62	269.77	5.56
	500	300	99.44	100	0.53	0.8	99.2	96.64	97.79	260.97	8.33
Spam Email	400	200	95.56	97.5	4.53	3.8	96.2	75.73	84.56	177.28	11.11
	400	300	95.57	97.5	4.52	3.8	96.2	75.77	84.58	170.38	11.11
	500	200	95.55	97	4.57	3.6	96.4	75.52	84.53	189.40	13.89
	500	300	95.53	97	4.60	3.6	96.4	95.40	84.45	197.53	13.89

4.4.1 Experiment 1: filter-based techniques

Experimental results for the proposed filter-based techniques are presented in this section. Specifically, this section reports the results obtained from the experiments performed on spam email, phishing email, credit card fraud and UCI datasets.

4.4.1.1 Results and discussion for spam email detection

The proposed filter-based techniques are validated on two different spam email datasets. As shown in Table 4.4, the first dataset (Dataset A) contains 4000 emails (500 spam email and 3500 legitimate emails). The second dataset (Dataset C) contains 4600 emails (1813 spam email and 2787 legitimate emails). As shown in Tables 4.8 – 4.14, the proposed filter-based techniques obtained a predictive accuracy of over 95% in less than 70 seconds. Also, the proposed boundary detection algorithms (i.e. ACOISA and EDISA) performed slightly slower than the NI-based techniques (i.e. FFISA, FPISA, SSISA, CSISA and BISA). The NI-based techniques were executed within 43 seconds, while the boundary detection algorithms were executed within 71 seconds. The difference in training speed is because of the additional tasks the boundary detection algorithms perform. The NI-based techniques perform only instance selection, while the boundary detection algorithms perform two tasks: boundary detection and instance selection. Furthermore, as shown in the result, storage requirement for all the proposed filter-based techniques is negligible. The boundary detection algorithms reduced the model size by over 91%, without significantly affecting the classification quality. Moreover, the NI-based techniques reduced the model size by over 90% and simultaneously improved the classification quality. In other words, as shown in Tables 4.8 – 4.14, the boundary detection techniques require a maximum of 8.33% of the training dataset, and the NI-based techniques require a maximum of 9.2% of the training dataset to produce robust classification models.

Moreover, the proposed techniques achieved a FP rate of less than 4% and a FN rate of less than 7%. The FP and FN rate indicate that the proposed techniques correctly classified about 96% legitimate emails and 93% spam emails. Although, standard SVM produced slightly better FP and FN rate, the proposed techniques obtained better training speed and storage reduction in all cases. Obviously, the improved classification speed came at the expense of FP and FN rate. The FP and FN rate can be further improved by training the model on additional features. As shown in Tables 4.8 and 4.9, the proposed techniques consume a small amount of space during classification. The boundary detection algorithms requires a maximum of 300 boundary instances and a subset size of 500 to produce improved classification models. Also, as shown in Tables 4.10 – 4.14, the NI-based techniques require a maximum of 700 instances to produce good training speed and predictive accuracy.

The best results obtained by the proposed techniques are benchmarked against the best results obtained by standard SVM, CLUS [1], KNN [8]. Table 4.17 and Figure 4.3 show the results of the comparison. As shown, the proposed techniques reduced the training data size by an average of 90% and slightly affected the predictive accuracy by a negligible value of 0.004%. However, there is a balanced trade-off between the predictive accuracy and training speed. Furthermore, as shown in the table, the proposed techniques improved SVM training speed by over 93%. Specifically, EDISA, ACOISA, FFISA, FPISA, SSISA, CSISA and BISA improved SVM training speed by 94.06%, 93.70%, 93.89%, 93.44%, 93.43%, 95.84% and 95.17% respectively. The improvement shows that the proposed filter-based techniques are fast and accurate techniques for instance selection.

In addition, the proposed techniques are further validated on another spam email dataset and compared to five existing instance selection techniques: PSC [251], DROP 3 [206], DROP 5 [206], GCNN [252] and POC-NN [253]. As shown in Table 4.18, the proposed techniques exceed the performance of all the compared techniques. The proposed techniques outperformed the compared techniques, in terms of classification speed and predictive accuracy. As shown in Table 4.18, the proposed filter-based techniques obtained a speed improvement of over 43%, 97%, 95%, 69%, 85%, when compared to PSC, DROP 3, DROP 5, GCNN and POC-NN, respectively. Moreover, the techniques obtained an accuracy improvement of 21.44%, 10.54%, 10.14%, 15.19% and 13.07%, when compared to PSC, DROP 3, DROP 5, GCNN and POC-NN, respectively. Generally, as shown in all the results, the proposed techniques are good instance selection and spam email detection techniques.

4.4.1.2 Results and discussion for phishing email detection

Table 4.8 – 4.14 shows the phishing email results for the proposed filter-based techniques. As shown in the tables, the proposed techniques produced excellent classification speed and accuracy. They all correctly classify over 99% phishing emails within a short time period. The NI-based techniques were executed in less than 60 seconds and the boundary detection algorithms were

executed in less than 72 seconds. Also, the NI-based techniques reduced the training data storage space by over 90% and the boundary detection algorithms reduced the training data storage space by 91.67%. Furthermore, the proposed techniques produced a FP rate of less than 1% and a FN rate of less than 3%. This shows that the proposed filter-based techniques correctly classified almost all of the phishing and legitimate emails. Additionally, as shown in Tables 4.8 and 4.9, EDISA produced the best results when K is 200 and the subset size is 400. Also, ACOISA produced its best results when K is 200 and the subset size is 500. This implies that both techniques require a maximum of 200 boundary instances to produce excellent classification models. As shown in Tables 4.10 – 4.14, all the NI-based techniques require a maximum of 700 instances to produce excellent results.

Furthermore, the best result produced by the proposed techniques are compared to standard SVM and two existing instance selection algorithms: CLUS [1] and KNN [8]. As shown in Table 4.16 and Figure 4.2, the proposed techniques outperformed CLUS [1] and KNN [8], and also improved SVM classification speed by over 95%. Specifically, EDISA, ACOISA, FFISA, FPISA, SSISA, CSISA and BISA improved SVM classification speed by 98.09%, 95.25%, 94.82%, 94.35%, 95.85%, 96.76%, 95.16%, respectively. The good results produced by the proposed filter-based techniques demonstrate their effectiveness in speed optimization and phishing email classification.

4.4.1.3 Results and discussion for credit card fraud detection

A different set of experiments was performed in order to test the performance of the proposed techniques on credit card fraud. Tables 4.8 – 4.14 report the experimental results produced by the techniques. As reported in the tables, the proposed techniques correctly classified over 96% credit card transactions in less than 90 seconds. Moreover, the NI-based techniques were executed in less than 90 seconds and require a maximum storage space of less than 7.5% of the training dataset. Also, the boundary detection algorithms were executed in less than 90 seconds and required a storage space of less than 7% of the training dataset. As shown in Tables 4.8 – 4.14, the NI-based techniques require a maximum of 700 instances to produce fast classification models. Also, the boundary detection algorithms require a maximum of 300 boundary instances to produce fast classification models. This implies that the proposed techniques require a small amount of storage space for training.

Table 4.5 shows the credit card fraud results for standard SVM. As shown in Table 4.15 and Figure 4.1, the proposed techniques improved SVM classification speed by over 97%, without meaningfully affecting SVM classification accuracy. Precisely, EDISA, ACOISA, FFISA, FPISA, SSISA, CSISA and BISA improved SVM classification speed by 97.12%, 95.71%, 96.02%, 96.74%, 97.40%, 98.34%, 95.91% respectively. Also, the results shows that the proposed filter-based techniques are faster than CLUS [1] and KNN [8]. Overall, the results reveal that the proposed techniques are fast and accurate techniques for SVM speed optimization and credit card fraud detection.

4.4.1.4 Results and Discussion for UCI datasets

The robustness of the proposed filter-based techniques are further demonstrated by validating them on 20 datasets provided by UCI dataset repository. UCI ML repository [249] consist of many widely used datasets, provided for experimental evaluation of ML algorithms. Table 4.19 shows the average predictive accuracy and time (in seconds) produced by the proposed techniques and standard SVM. In the table, for each dataset, the best three training speed are underlined. As shown in the table, the proposed techniques consistently produced better training speed in 100% of the datasets (20 out of 20) used for evaluation, without significantly affecting SVM predictive accuracy. Moreover, results shows that the CSISA produced the best training speed in most cases, followed by FPISA. Also, the result shows that the proposed techniques outperform standard SVM, in terms of speed-accuracy trade-off.

The proposed techniques are further compared to three existing filter-based instance selection techniques: Wilson [254], RT3 [255] and ICF [256]. Table 4.20 shows the results of the comparison. The best predictive accuracy for each of the datasets is underlined. As shown in the table, the proposed techniques outperform the three compared techniques in 69% (9 out of 13) of the datasets used for evaluation. The results show that the proposed filter-based techniques can also be applied to other classification problems, different from e-fraud.

Table 4.8: Filter-based EDISA results for e-fraud detection

Mail Type	K	Subset Size	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage Reduction
Credit Card Fraud	200	400	98.11	98.80	0.06	18.8	81.20	99.34	89.3	59.97	4.44
	300	400	97.94	98.80	0.44	17.10	82.90	96.3	88.82	82.95	6.67
	200	500	97.90	98.80	0.39	17.93	82.07	97.15	88.64	65.33	4.44
	300	500	98.10	98.8	0.18	17.82	82.18	98.27	89.37	87.93	6.67
Phishing Email	200	400	99.41	100	0.42	1.78	98.22	97.22	97.59	18.00	5.56
	300	400	99.38	100	0.46	1.78	98.22	96.99	97.46	24.2	8.33
	200	500	99.40	100	0.40	2.00	98.00	97.35	97.53	19.82	5.56
	300	500	99.40	100	0.40	2.00	98.00	97.35	97.53	19.82	8.33
Spam Email	200	400	96.51	97.50	3.16	5.84	94.16	81.03	87.07	35.15	5.56
	300	400	96.42	97.50	3.14	6.66	93.34	80.95	86.68	59.59	8.33
	200	500	96.61	97.50	3.29	4.14	95.86	80.70	97.56	35.49	5.56
	300	500	96.63	97.50	3.29	3.96	96.04	80.73	87.66	56.60	8.33

Table 4.9: Filter-based ACOISA results for e-fraud detection

Mail Size	K	Subset Size	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage Reduction
Credit Card Fraud	200	400	96	99.2	3.08	12.3	87.7	79.54	82.2	61.19	4.44
	300	400	96.63	99.2	2.55	10.86	89.14	81.96	84.51	88.84	6.67
	200	500	96.52	99.2	2.29	14.45	85.55	84.64	83.9	66.15	4.44
	300	500	96.03	98.8	3.21	10.96	89.04	78.63	82.42	93.43	6.67
Phishing Email	200	400	99.2	100	0.62	2.08	97.92	96.13	96.81	38.46	5.56
	300	400	99.46	100	0.33	2.04	97.96	97.79	97.75	64.35	8.33
	200	500	99.35	100	0.43	2.18	97.82	97.13	97.36	44.76	5.56
	300	500	99.33	100	0.49	1.92	98.08	96.9	97.32	71.08	8.33
Spam Email	200	400	96.53	98.75	3.4	3.98	96.02	80.33	87.39	39.81	5.56
	300	400	96.36	97.50	3.53	4.42	96.00	95.58	86.79	57.11	8.33
	200	500	96.56	97.5	3.36	4.04	95.96	80.49	87.46	38.99	5.56
	300	500	96.54	97.5	3.41	3.82	96.18	80.31	87.44	60.08	8.33

Table 4.10: Filter-based FFISA results for e-fraud detection

Mail Type	NP	Subset Size	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage Reduction
Credit Card Fraud	5	500	96.94	99.20	2.00	12.85	87.15	85.29	85.40	50.05	5.28
	5	700	97.24	99.20	1.84	11.24	88.76	85.40	86.51	82.52	7.43
	10	500	96.99	99.20	1.91	13.02	86.98	85.47	85.49	49.42	5.17
	10	700	97.08	99.20	1.97	11.64	88.36	84.91	85.98	82.22	7.31
Phishing Email	5	500	99.32	100	0.45	2.30	97.70	97.22	97.28	34.23	6.61
	5	700	99.47	100	0.29	2.22	97.78	98.08	97.8	45.95	9.34
	10	500	99.31	100	0.46	2.30	97.70	97.24	97.26	31.79	6.50
	10	700	99.37	100	0.41	2.16	97.84	97.42	97.45	42.99	9.17
Spam Email	5	500	96.25	97.50	3.70	4.08	95.92	79.11	86.56	41.47	6.60
	5	700	96.45	97.50	3.43	4.34	95.66	79.95	86.99	58.26	9.30
	10	500	96.16	97.50	3.81	4.04	95.96	78.64	86.30	43.88	6.49
	10	700	96.31	99.25	3.64	4.02	95.98	79.38	86.76	62.90	9.20

Table 4.11: Filter-based FPISA results for e-fraud detection

Mail Type	NP	Subset Size	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage Reduction
Credit Card Fraud	5	500	97.18	99.00	1.73	12.69	87.31	86.22	86.12	47.08	5.08
	5	700	97.22	99.20	1.83	11.75	88.25	85.90	86.50	67.61	7.20
	10	500	96.95	99.00	1.95	12.99	87.01	85.09	85.23	53.05	4.98
	10	700	97.14	99.20	1.87	12.04	87.96	85.61	86.16	76.16	7.10
Phishing Email	5	500	99.34	100	0.42	2.32	97.68	97.29	97.32	28.54	6.34
	5	700	99.45	100	0.32	2.20	97.80	97.91	97.71	48.82	9.01
	10	500	99.16	100	0.65	2.24	97.76	96.18	96.71	28.80	6.25
	10	700	99.33	100	0.49	1.96	98.04	96.98	97.33	42.70	8.86
Spam Email	5	500	96.27	97.50	3.49	5.48	94.52	80.14	86.10	43.84	6.38
	5	700	96.41	97.50	3.55	3.90	96.10	97.70	87.02	62.57	9.00
	10	500	95.96	97.50	3.93	4.84	95.16	78.11	85.43	45.21	6.21
	10	700	96.21	97.50	3.69	4.48	95.52	79.16	86.26	62.76	8.87

Table 4.12: Filter-based SSISA results for e-fraud detection

Mail Type	**NP**	**Subset Size**	**APA(%)**	**GB(%)**	**FP(%)**	**FN(%)**	**R(%)**	**Pr(%)**	**FM(%)**	**T(s)**	**Storage Reduction**
Credit Card Fraud	5	500	97.24	99.00	1.68	12.59	87.41	86.65	86.34	53.99	5.24
	5	700	97.16	99.00	1.93	11.23	88.77	84.83	86.21	67.56	7.39
	10	500	97.18	99.20	1.66	13.59	86.41	87.05	86.01	57.52	5.16
	10	700	97.18	99.00	1.89	11.43	88.57	85.42	86.35	67.34	7.30
Phishing Email	5	500	99.34	100	0.45	2.20	97.80	97.32	97.36	38.33	6.51
	5	700	99.44	100	0.32	2.24	97.76	97.95	97.70	53.27	9.22
	10	500	99.32	100	0.45	2.30	97.70	97.22	97.28	38.73	6.39
	10	700	99.44	100	0.32	2.24	97.76	97.83	97.67	59.51	9.12
Spam Email	5	500	96.26	97.50	3.67	4.26	95.74	79.27	86.54	45.61	6.53
	5	700	96.39	97.50	3.51	4.32	95.68	80.03	86.84	63.49	9.24
	10	500	96.37	97.50	3.59	3.94	96.06	79.56	86.92	45.75	6.38
	10	700	96.45	97.50	3.51	3.82	96.18	79.90	87.18	70.90	9.11

Table 4.13: Filter-based CSISA results for e-fraud detection

Mail Type	NP	Subset Size	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage Reduction
Credit Card Fraud	5	500	96.85	99.00	1.98	13.91	86.09	84.93	84.59	32.85	3.28
	5	700	96.84	98.60	2.09	13.04	86.96	84.48	84.90	45.92	4.65
	10	500	96.94	99.20	1.87	14.04	85.96	85.54	84.97	34.72	3.18
	10	700	96.86	98.80	2.07	12.95	87.05	84.53	84.95	49.48	4.57
Phishing Email	5	500	99.07	100	0.73	2.36	97.64	95.76	96.40	18.14	4.06
	5	700	99.31	100	0.45	2.34	97.66	97.06	97.22	30.54	5.90
	10	500	99.19	100	0.56	2.54	97.46	96.70	96.84	21.69	3.99
	10	700	99.26	100	0.51	2.30	97.70	96.97	97.11	30.80	5.73
Spam Email	5	500	96.20	100	3.74	4.24	95.76	78.93	86.38	29.89	4.06
	5	700	96.16	97.50	3.75	4.48	95.52	78.96	86.21	41.74	5.80
	10	500	95.98	99.25	3.88	4.98	95.02	78.44	85.55	30.66	4.02
	10	700	96.31	97.50	3.59	4.40	95.60	79.58	86.60	39.63	5.73

Table 4.14: Filter-based BISA results for e-fraud detection

Mail Type	NP	Subset Size	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage Reduction
Credit Card Fraud	5	500	97.13	99.20	1.80	12.78	87.22	86.46	86.11	53.14	5.10
	5	700	97.03	99.20	2.11	10.91	89.09	83.53	85.71	74.30	7.26
	10	500	96.75	99.00	2.30	11.97	88.03	82.65	84.64	52.41	5.02
	10	700	97.40	99.20	1.56	12.14	87.86	87.43	87.07	84.88	7.15
Phishing Email	5	500	99.29	100	0.49	2.26	97.74	97.01	97.18	30.82	6.43
	5	700	99.39	100	0.38	2.20	97.80	97.57	97.52	43.64	9.05
	10	500	99.42	100	0.37	2.08	97.92	97.70	97.65	30.02	6.27
	10	700	99.43	100	0.33	2.28	97.72	97.79	97.62	45.62	8.92
Spam Email	5	500	96.15	100	3.80	4.22	95.78	78.75	86.27	44.87	6.38
	5	700	96.29	97.50	3.65	4.08	95.92	79.24	86.66	61.97	9.10
	10	500	96.36	97.50	3.57	4.14	95.86	79.52	86.84	46.03	6.27
	10	700	96.25	97.50	3.71	4.10	95.90	79.11	86.55	61.09	8.95

Table 4.15: Filter-based Techniques vs. other techniques for credit card fraud

Technique	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage Reduction
EDISA	98.11	98.8	0.06	18.8	81.2	99.34	89.3	59.97	4.44
ACOISA	96.63	99.2	2.55	10.86	89.14	81.96	84.51	88.84	6.67
FFISA	97.24	99.20	1.84	11.24	88.76	85.40	86.51	82.52	7.43
FPISA	97.22	99.20	1.83	11.75	88.25	85.90	86.50	67.61	7.20
SSISA	97.24	99.00	1.68	12.59	87.41	86.65	86.34	53.99	5.24
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 **[1]**	98.47	99.4	0.46	11.31	88.69	95.48	91.9	684.06	41.67
KNN **[8]**	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 4.16: Filter-based techniques vs. other techniques for phishing email

Technique	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage Reduction (%)
EDISA	99.41	100	0.42	1.78	98.22	97.22	97.59	18.00	5.56
ACOISA	99.35	100	0.43	2.18	97.82	97.13	97.36	44.76	5.56
FFISA	99.47	100	0.29	2.22	97.78	98.08	97.8	45.95	9.34
FPISA	99.45	100	0.32	2.20	97.80	97.91	97.71	48.82	9.01
SSISA	99.44	100	0.32	2.24	97.76	97.95	97.70	53.27	9.22
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 **[1]**	99.53	100	0.23	2.16	97.84	98.47	98.03	337.46	41.67
KNN **[8]**	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 4.17: Filter-based techniques vs. other techniques for spam email

Technique	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage Reduction (%)
EDISA	96.63	97.5	3.29	3.96	96.04	80.73	87.66	56.60	8.33
ACOISA	96.54	97.5	3.41	3.82	96.18	80.31	87.44	60.08	8.33
FFISA	96.45	97.50	3.43	4.34	95.66	79.95	86.99	58.26	9.30
FPISA	96.41	97.50	3.55	3.90	96.10	97.70	87.02	62.57	9.00
SSISA	96.49	97.50	3.45	3.94	96.06	80.16	87.29	62.67	9.12
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 [1]	96.44	100	2.61	10.28	89.72	84.41	85.35	311.98	41.67
KNN [8]	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 4.18: Filter-based proposed techniques vs other techniques for spambase dataset

Technique	**APA(%)**	**T(s)**
EISA	87.92	96.87
ACOISA	84.38	99.22
FFISA	88.37	107.88
FPISA	88.11	98.12
SSISA	87.96	102.30
CSISA	86.71	55.32
BISA	88.15	91.22
PSC **[251]**	71.95	189.57
DROP 3 **[206]**	78.44	3782.57
DROP 5 **[206]**	78.72	2226.42
GCNN **[252]**	73.54	348.56
POC-NN **[253]**	75.37	735.08

Table 4.19: Filter-based proposed techniques vs standard SVM

Dataset Name	**FFISA**		**FPISA**		**SSISA**		**CSISA**		**BISA**		**EDISA**		**ACOISA**		**SVM**	
	Accr	**Time**	**Accr**	**Time**	**Accr**	**Time**	**Accr**	**Time**	**Accr**	**Time**	**Accr**	**Time**	**Accr**	**Time**	**Accr**	**Time**
Abalone	53.37	65.88	53.27	71.94	53.14	66.24	52.96	42.28	53.21	65.77	51.26	92.54	51.09	126.6	55.71	2010
Balance Scale	90.82	48.72	90.65	47.18	90.65	50.55	88.71	29.82	90.52	49.64	88.52	36.07	88.23	38.22	93.71	101.1
Breast Tissue	57.60	7.19	60.70	7.01	58.40	7.21	58.10	7.24	57.70	7.20	57.7	13.32	62.4	12.61	64.6	15.98
Bupa	66.50	25.64	65.44	25.29	66.21	23.63	62.21	14.89	66.56	25.52	67.79	40.02	65.24	40.61	71.56	64.81
credit-g	74.34	78.49	74.07	78.03	74.05	86.22	71.63	50.33	73.55	83.14	74.04	69.06	72.31	73.53	75.95	299.9
Cleaveland	61.10	20.39	60.72	19.60	61.38	21.01	59.59	12.02	60.90	19.56	62.83	31.25	52.03	43.15	63.21	53.55
Ecoli	86.09	24.05	85.27	25.21	86.27	26.86	84.06	16.41	85.82	24.61	86.64	35.3	84.97	38.38	87.36	62.1
Glass	64.95	16.38	64.90	15.35	65.14	15.52	61.05	9.42	63.52	14.82	60.14	22.56	64.57	20.56	65.67	33.95
Hungarian	63.28	23.31	63.83	21.75	62.90	23.60	63.34	14.69	64.03	24.20	65.69	17.88	56.34	40.45	63.86	52.12
Iris	94.80	10.61	95.60	9.65	95.27	9.83	94.47	6.14	94.73	9.53	95.13	15.92	96.13	17.05	95.5	21.45
Liver	65.56	28.56	66.00	25.46	65.44	27.03	62.56	15.96	65.24	27.58	68.88	33.89	66.74	35.06	72.47	58.26
Pima Indians	75.36	74.42	75.18	66.06	75.63	72.54	74.25	42.91	75.03	70.93	74.74	33.57	71.16	33.64	76.92	126.7
Post Operative	70.88	7.37	71.00	7.13	71.25	7.56	71.63	6.90	71.25	7.04	71.13	6.67	65.25	8.36	71.25	11.87
Transfusion	78.32	66.17	78.55	46.28	77.91	66.82	77.74	32.06	78.09	53.35	75.99	32.8	70.35	33.47	78.61	135.2
Vertebral-3c	83.13	24.03	83.00	25.45	82.68	25.35	82.29	14.82	84.16	21.5	78.58	19.85	81.32	21.00	85.61	53.51
Voting	94.98	34.31	94.79	33.37	95.09	29.98	94.09	18.73	94.93	30.86	88.88	34.24	91.49	42.13	95.77	83.07
Waveform	84.09	78.46	83.92	76.25	84.13	79.72	82.81	51.03	83.89	79.05	81.81	132.2	85.26	126.9	86.98	2501
Wine	97.47	8.50	97.06	8.39	96.53	9.04	96.29	5.29	97.59	8.75	95.35	12.11	93.71	13.24	97.47	32.58
Yeast	57.68	67.77	57.25	67.19	57.14	73.52	55.48	50.06	57.39	80.07	52.48	56.07	52.42	58.25	59.45	306
Zoo	92.00	7.19	92.50	7.12	92.00	7.24	90.40	7.32	91.40	7.66	94	17.27	91.3	7.16	95	17.74
Average	**75.62**	**37.02**	**75.25**	**31.55**	**75.17**	**33.34**	**73.71**	**20.64**	**75.13**	**32.66**	**74.58**	**37.63**	**73.11**	**41.52**	**77.83**	**302.04**

Table 4.20: Filter-based techniques vs wilson, RT3 and ICF

Dataset Name	**FFISA**	**FPSIA**	**SSISA**	**CSISA**	**BISA**	**EDISA**	**ACOISA**	**Wilson [254]**	**RT3 [255]**	**ICF [256]**
	Accr	**Accr**	**Accr**	**Accr**	**Accr**	**Accr**	**Accr**	**Accr**	**Accr**	**Accr**
Abalone	53.37	53.27	53.14	52.96	53.21	51.26	51.09	22.01	22.11	22.74
Balance S	90.82	90.65	90.65	88.71	90.52	88.52	88.23	86.04	83.4	81.47
Bupa	66.50	65.44	66.21	62.21	66.56	67.79	65.24	61.81	61.23	60.75
Ecoli	86.09	85.27	86.27	84.06	85.82	86.64	84.97	86.27	82.84	81.34
Glass	64.95	64.90	65.14	61.05	63.52	60.14	64.57	69.05	69.05	69.64
Hungarian	63.28	63.83	62.90	63.34	64.03	65.69	56.34	79.91	80.17	78.3
Iris	94.80	95.60	95.27	94.47	94.73	95.13	96.13	95.33	93.61	92.56
Pima Ind	75.36	75.18	75.63	74.25	75.03	74.74	71.16	71.27	71.08	69.17
Post Opr	70.88	71.00	71.25	71.63	71.25	71.13	65.25	66.94	69.44	65.28
Voting	94.98	94.79	95.09	94.09	94.93	88.88	91.49	93.28	93.77	91.19
Wavefrm	84.09	83.92	84.13	82.81	83.89	81.81	85.26	76.62	76.14	73.93
Wine	97.47	97.06	96.53	96.29	97.59	95.35	93.71	86.43	86.43	83.81
Zoo	92.00	92.50	92.00	90.40	91.40	94	91.3	96.25	87.08	92.42
Average	**79.58**	**79.49**	**79.55**	**78.17**	**79.42**	**78.54**	**77.29**	**76.25**	**75.10**	**74.04**

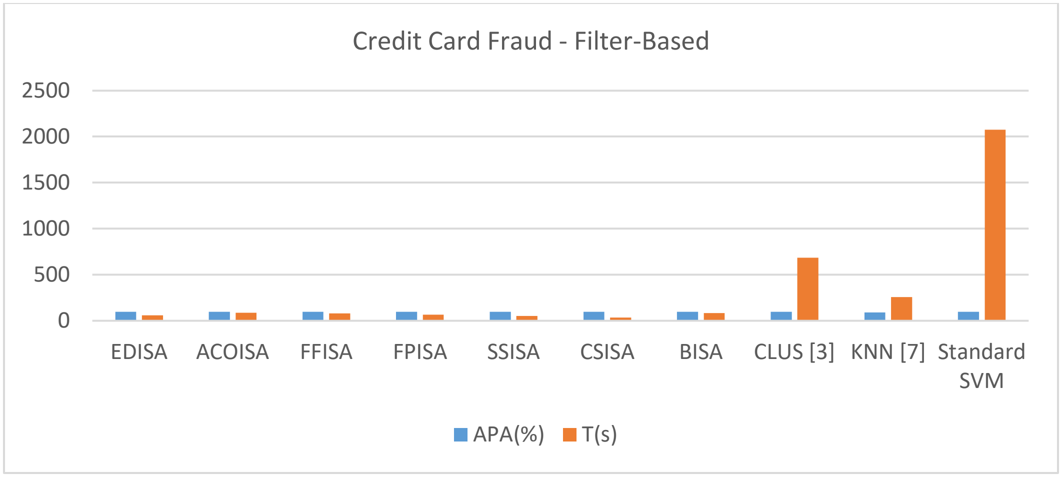

Figure 4.1: Filter-based Technique vs. CLUS, KNN and Standard SVM - Credit Card Fraud

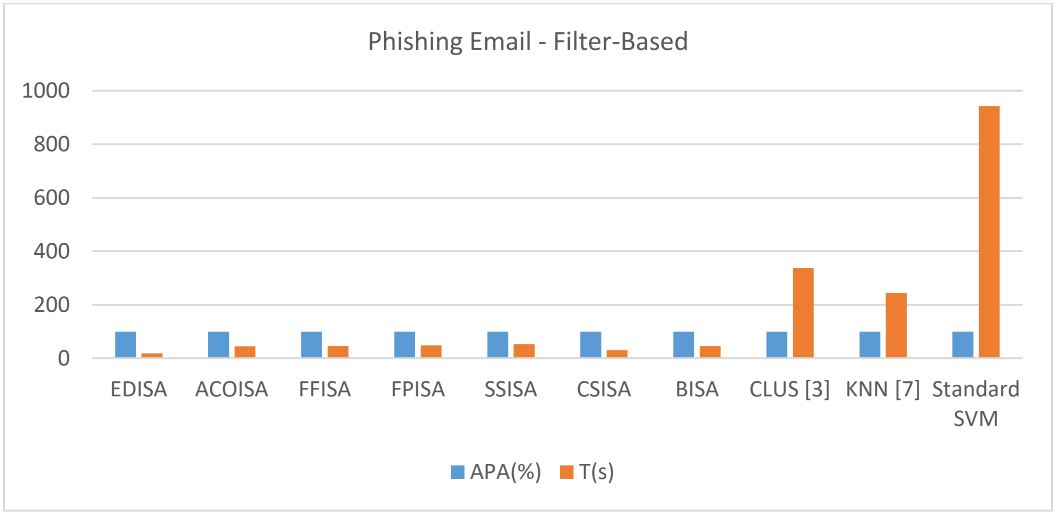

Figure 4.2: Filter-based technique vs. CLUS, KNN and standard SVM - phishing email

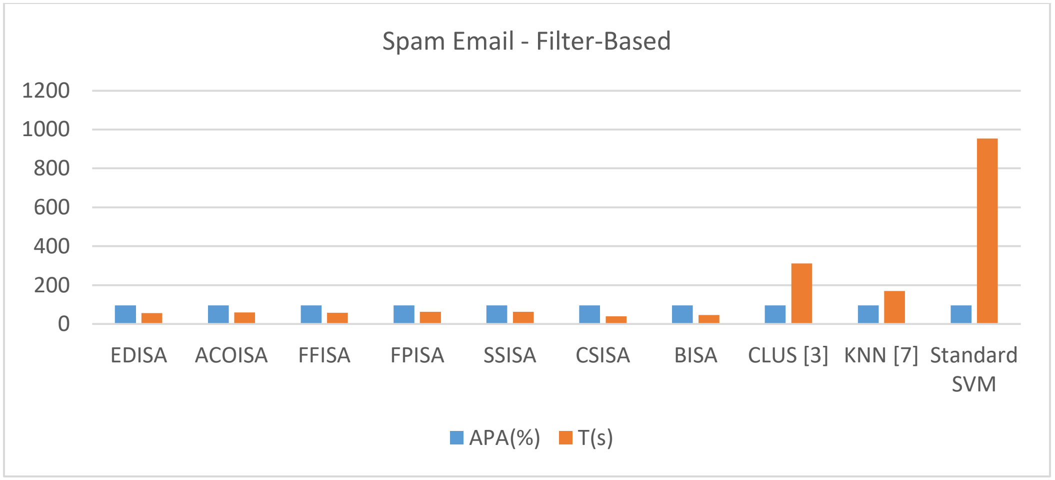

Figure 4.3: Filter-based technique vs. CLUS, KNN and standard SVM - spam email

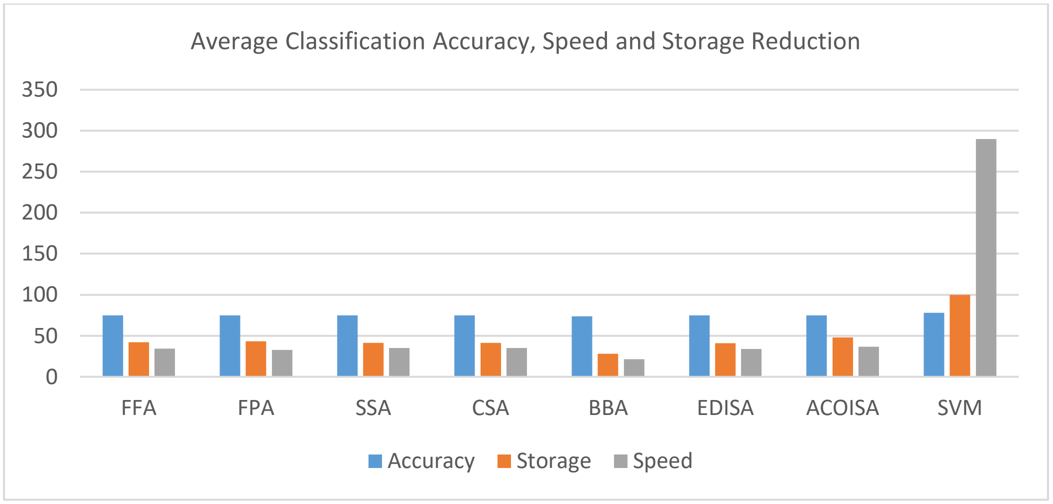

Figure 4.4: Filter-based technique vs. standard SVM (UCI datasets)

4.4.2 Experiment 2: wrapper-based techniques

This thesis proposes five wrapper-based instance selection techniques for improving SVM predictive accuracy. The five techniques are validated on 23 datasets containing spam emails, phishing emails, credit card fraud and 20 other problems. As mentioned in Section 4.1, unlike the filter-based techniques that searches through a subset of the dataset, the wrapper-based techniques are designed to search through the entire training data for relevant instances. Tables 4.21 – 4.25 report the results for the wrapper-based techniques, while Figures 4.5 – 4.8 show the comparisons between standard SVM and the wrapper-based techniques. Table 4.26 shows the results for credit card fraud, Table 4.27 shows the results for phishing emails and Table 4.28 shows the results for spam emails. For each table, the best three predictive accuracy are underlined. As shown in the results, the wrapper-based techniques improved SVM predictive accuracy for most cases. They also simultaneously improved SVM training speed, especially for large datasets.

4.4.2.1 Result and discussion for spam email

Tables 4.21 – 4.25 reports the average predictive accuracy, global best, FP rate, FN rate, time (in seconds) and storage reduction for the experiments performed on spam emails. Global best refers to the best predictive accuracy achieved during the cross validation of each dataset. As shown in the tables, FFISA obtained the least predictive accuracy of 96.67% and CSISA obtained the best predictive accuracy of 96.92%. FPISA, SSISA and BISA obtained predictive accuracies of 96.75%, 96.80% and 96.80%, respectively. This implies that the five proposed wrapper-based techniques correctly classified an average of over 96% of the entire datasets. Moreover, CSISA achieved the highest global best solution of 99.25%, followed by FPISA and BISA. This implies that CSISA correctly classified over 99% of the dataset during some rounds of cross-validation. Moreover, CSISA obtained the best FP rate of 2.89%, followed by FPISA (2.96%) and FFISA (3.06%). SSISA and BISA obtained a FP rate of 3.08% and 3.11%, respectively. The FP rate shows that CSISA, FPISA, FFISA, SSISA and BISA correctly classified 97.11%, 97.04%, 96.94%, 96.92% and 96.89% legitimate emails, respectively. Furthermore, BISA obtained the best FN rate of 3.86%, followed by SSISA (4.02%) and CSISA (4.40%). The FN rate shows that the proposed wrapper-based techniques correctly classified approximately 96% of spam emails.

As shown in Table 4.28, CSISA obtained the best storage reduction. It reduced the training dataset by over 54%. FFISA obtained the next best storage reduction, followed by FPISA, BISA and SSISA. They all reduced the training dataset to approximately half its size. Moreover, BISA obtained the best training speed of 442.32 seconds, followed by CSISA (454.91 seconds). SSISA obtained the lowest training speed of 611.98 seconds. Although, the training speed of the wrapper-based techniques is not very high (compared to the filter-based techniques), they still perform at a faster rate than the standard SVM. However, as aforementioned, the primary objective of the wrapper-based techniques is to improve the predictive accuracy of SVM. Overall, CSISA produced the best results, when compared to the other filter-based techniques.

The performances of the proposed wrapper-based techniques are compared to the performances of standard SVM. Table 4.29 shows the results of the comparison for the spam email dataset. As shown, the five proposed techniques achieved better predictive accuracy, compared to the standard SVM. They also outperform the standard SVM in terms of training speed. Specifically, FFISA, FPISA, SSISA, CSISA and BISA improved SVM training speed by 32.03%, 30.22%, 28.27%, 46.68% and 48.15%, respectively. The speed improvement is particularly obvious for large datasets, which underscores the importance of SVM speed optimization. Furthermore, the five proposed techniques produced better FP rate, compared to the standard SVM. Also, BISA, CSISA and SSISA outperform standard SVM, in terms of FN rate. The improved performance of the proposed wrapper-based techniques indicates that the proposed techniques are better instance-selection techniques, compared to the standard SVM. The improved results also show that NI algorithms are good instance-selection techniques.

4.4.2.2 Result and discussion for phishing email

Tables 4.21 – 4.25 show the average predictive accuracy, global best predictive accuracy, FP rate, FN rate, time (in seconds) and storage reduction for the experiments performed on phishing emails. As shown in the table, the proposed techniques correctly classified over 99.6% of the dataset. FPISA produced the best predictive accuracy of 99.63%. SSISA, CSISA and BISA produced the same predictive accuracy of 99.62%. FFISA produced a predictive accuracy of 99.60%. The high predictive accuracy can be attributed to the quality of the extracted features used to train the classifier. Furthermore, as shown in the result, the proposed wrapper-based techniques produced a global best predictive accuracy of 100%. This shows that the proposed wrapper-based techniques correctly classified 100% of the dataset in some rounds of cross-validation, which demonstrate the efficacy of the proposed techniques. Moreover, FPISA produced the best FP rate of 0.11%, followed by CSISA (0.13%) and BISA (0.13%). The good FP rate shows that the proposed wrapper-based techniques classified virtually all legitimate emails correctly. Furthermore, SSISA produced the best FN rate of 2.08%, followed by FFISA (2.10%) and BISA (2.14%). CSISA and FPISA produced a FN rate of 2.18% and 2.20%, respectively. The FN rate indicates that the proposed techniques classified approximately 98% of phishing emails correctly. Specifically, SSISA correctly classified 97.92% phishing emails, while FFISA, BISA, CSISA and FPISA correctly classified 97.90%, 97.86%, 97.82% and 97.80% phishing emails, respectively.

Tables 4.21 – 4.25 show the storage reduction and the training speed produced by the proposed wrapper-based techniques. As shown in the tables, the proposed techniques reduced the training dataset size by an average of 50% and simultaneously improved SVM training speed by over 56%. CSISA produced the best storage reduction percentage of 47.83%, followed by FPISA (49.82%) and BISA (50.10%). The storage reduction indicates that the proposed wrapper-based techniques require approximately half (that is, 50%) of the training dataset to produce robust classification models. Although the primary objective of the proposed wrapper-based techniques is to improve SVM classification accuracy, the techniques also simultaneously improved SVM training speed. As shown in the tables, FFISA achieved the best training speed of 371.38 seconds and improved SVM training speed by 60.62%. Moreover, CSISA and SSISA produced the next best training speed and improved SVM training speed by 59.91% and 58.11%, respectively. FPISA and BISA improved SVM training speed by 57% and 56.57%, respectively.

The results produced by the proposed wrapper-based techniques are compared to the results produced by standard SVM. As shown in Table 4.30, although the proposed techniques slightly reduced SVM predictive accuracy by a non-significant value of 0.06%, they improved SVM training speed by over 56% and also reduced the training data size by an average of 50%. The excellent predictive accuracy and speed improvement produced by the proposed wrapper-based techniques indicate their superiority over the standard SVM, in terms of speed-accuracy trade-off. Generally, CSISA produced the best results for phishing emails, compared to the other proposed wrapper-based techniques.

4.4.2.3 Result and discussion for credit card fraud

Tables 4.21 – 4.25 show the results produced by the wrapper-based techniques for credit card fraud. As shown in the tables, the proposed wrapper-based techniques correctly classified over 98% of credit card transactions. The tables show that SSSIA produced the best predictive accuracy of 98.93%, followed by FPISA (98.86%) and BISA (98.84%). The good predictive accuracy underscores the generalization performance of the proposed wrapper-based techniques. Moreover, the table shows that the proposed techniques produced a global best predictive accuracy of 99.60%. This indicates that the proposed techniques correctly classified 99.6% of credit card transactions in at least one round of cross-validation. This shows the robustness of the models produced by the proposed wrapper-based techniques. Moreover, the proposed wrapper-based techniques achieved a FP rate of less than 0.4%. FPSIA produced the best FP rate of 0.20%, while SSISA and CSISA produced the second best FP rate of 0.26%. The improved FP rate indicates that the proposed wrapper-based techniques correctly classified nearly all of the legitimate credit card transactions. FPISA correctly classified over 99.8% of credit card fraud transactions, while SSISA and CSISA correctly classified 99.74% of credit card transactions. BISA and FFISA correctly classified 99.71% and 99.65% of credit card transactions, respectively.

Furthermore, Tables 4.21 – 4.25 show the FN rate produced by the proposed wrapper-based techniques. As shown in the table, SSISA produced the best FN rate of 8.44%, followed by FPISA (9.02%) and BISA (9.07%). The poor FN rates are primarily caused by the dataset quality used for evaluation. As mentioned in Section 1.5, the credit card fraud dataset used in this research was modified by the dataset owners, and its features were transformed to numerical values. Furthermore, FPISA produced the best training speed of 776.25 seconds, followed by BISA (828.73 seconds) and FFISA (883.82 seconds). Moreover, as shown in the results, the proposed techniques reduced the training dataset size by an average of 50%. CSISA produced the best storage reduction of 44.1%, while BISA and FFISA produced the second and third best storage reduction of 50.01% and 50.02%. The good storage reduction produced by the proposed wrapper-based techniques shows their usefulness in applications that process massive datasets with limited storage space.

Table 4.26 shows the comparison between the proposed techniques and the standard SVM. The best three examples of predictive accuracy are underlined. As shown in the table, the wrapper-based techniques produced better predictive accuracy compared to the standard SVM. Moreover, as shown in the result, the proposed techniques require approximately 50% of the training dataset to produce fast and accurate classification models for credit card fraud detection. Furthermore, Table 4.26 shows that the proposed wrapper-based techniques improved SVM training speed by over 54%. Specifically, FFISA, FPISA, SSISA, CSISA and BISA improved SVM training speed by 57.36%, 62.55%, 54.41%, 68.65% and 60.02%, respectively. Overall, SSISA produced the best result for credit card fraud, when compared to the other proposed wrapper-based techniques.

4.4.2.4 Results and discussion for UCI datasets

The wrapper-based techniques are further validated on 20 datasets provided by the UCI dataset repository. Table 4.29 shows the predictive accuracy, training speed and storage reduction percentage produced by the proposed wrapper-based techniques and standard SVM. For each dataset, the best three results are underlined. As shown in the table, the proposed wrapper-based techniques consistently outperform the standard SVM in 80% of the datasets (16 out of 20) used for evaluation. Moreover, as shown in the table, SSISA produced the best average predictive accuracy in most cases, followed by CSISA and FPISA. Moreover, CSISA produced the best average training speed, followed by FPISA and SSISA.

The proposed wrapper-based techniques are compared to an existing state-of-the-art wrapper-based instance selection technique (ADR-Miner [17]). Table 4.30 shows the result of the comparison. ADR-Miner was designed to use two classification algorithms for evaluation. One classification algorithm is used to evaluate the quality of each candidate solution and the second classification algorithm is used to build the final model. Ismail *et al.* [17] presented the results for different algorithm combinations. To ensure a fair comparison, we compare the wrapper-based techniques to the algorithm combination that used SVM at both the instance-selection stage and the model-construction stage. This is because the proposed techniques also used SVM at both stages. For each dataset, the best three results are underlined. As shown in the table, the five proposed wrapper-based outperform ADR-Miner in 90% (9 out of 10) of the dataset.

Table 4.21: Wrapper-based FFISA results for e-fraud detection

Mail Type	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage
Credit Card Fraud	98.77	99.60	0.35	9.35	90.65	96.76	93.53	883.82	50.02
Phishing Email	99.60	100	0.16	2.1	97.9	98.9	98.28	371.38	50.12
Spam Email	96.67	97.5	3.06	5.24	94.76	81.91	87.4	579.88	49.9

Table 4.22: Wrapper-based FPISA results for e-fraud detection

Mail Type	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage
Credit Card Fraud	98.86	99.60	0.2	9.02	90.98	97.33	93.96	776.25	50.14
Phishing Email	99.63	100	0.11	2.2	97.8	99.24	98.4	405.58	49.82
Spam Email	96.75	98.75	2.96	5.34	94.66	82.39	87.66	595.28	49.97

Table 4.23: Wrapper-based SSISA results for e-fraud detection

Mail Type	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage
Credit Card Fraud	98.93	99.6	0.26	8.44	91.56	97.78	94.37	945.17	53.64
Phishing Email	99.62	100	0.14	2.08	97.92	99.05	98.37	395.1	50.42
Spam Email	96.8	97.50	3.08	4.02	95.98	81.67	88.2	611.98	51.32

Table 4.24: Wrapper-based CSISA results for e-fraud detection

Mail Type	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage
Credit Card Fraud	98.83	99.60	0.26	9.57	90.43	97.54	93.75	649.95	44.1
Phishing Email	99.62	100	0.13	2.18	97.82	99.14	98.35	378.12	47.83
Spam Email	96.92	99.25	2.89	4.4	95.6	82.73	88.56	454.91	46.21

Table 4.25: Wrapper-based BISA results for e-fraud detection

Mail Type	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage
Credit Card Fraud	98.84	99.60	0.29	9.07	90.93	97.26	93.92	828.73	50.01
Phishing Email	99.62	100	0.13	2.14	97.86	99.1	98.37	409.69	50.1
Spam Email	96.8	97.75	3.11	3.86	96.14	81.56	88.21	442.32	50.03

Table 4.26: Wrapper-based techniques vs standard SVM for credit card

Technique	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage Reduction (%)
FFISA	98.77	99.60	0.35	9.35	90.65	96.76	93.53	883.82	50.02
FPISA	98.86	99.60	0.2	9.02	90.98	97.33	93.96	776.25	50.14
SSISA	98.93	99.6	0.26	8.44	91.56	97.78	94.37	945.17	53.64
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 4.27: Wrapper-based techniques vs standard SVM for phishing email

Technique	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage Reduction (%)
FFISA	99.60	100	0.16	2.1	97.9	98.9	98.28	371.38	50.12
FPISA	99.63	100	0.11	2.2	97.8	99.24	98.4	405.58	49.82
SSISA	99.62	100	0.14	2.08	97.92	99.05	98.37	395.1	50.42
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 4.28: Wrapper-based techniques vs standard SVM for spam email

Technique	APA(%)	GB(%)	FP(%)	FN(%)	R(%)	Pr(%)	FM(%)	T(s)	Storage Reduction (%)
FFISA	96.67	97.5	3.06	5.24	94.76	81.91	87.4	579.88	49.9
FPISA	96.75	98.75	2.96	5.34	94.66	82.39	87.66	595.28	49.97
SSISA	96.80	97.50	3.08	4.02	95.98	81.67	88.2	611.98	51.32
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 4.29: Wrapper-based proposed techniques vs standard SVM

Dataset Name	**FFISA**			**FPISA**			**SSISA**			**CSISA**			**BISA**			**Standard SVM**		
	Accr	**Stor**	**Time**	**Accr**	**Stor**	**Time**	**Accr**	**Stor**	**Time**	**Accr**	**Stor**	**Time**	**Accr**	**Stor**	**Time**	**Accr**	**Stor**	**Time**
Abalone	56.31	50.08	1176.15	56.73	49.95	974.62	56.76	53.02	1357.68	56.72	37.86	745.95	56.42	50.02	1014.01	55.71	0	2010
Balance Scale	91.31	50.23	86.01	91.32	50.26	83.03	91.98	53.09	83.73	91.52	42.26	73.98	91.47	50.5	81.91	93.71	0	101.1
Breast Tissue	68.3	51.09	13.83	69.9	50.88	13.85	71.3	53.26	14.47	69.5	47.41	14.23	67.6	50.66	13.81	64.6	0	15.98
Bupa	69.5	50.26	46.1	70.24	50.32	45.09	70.82	52.65	49.19	69.85	40.04	39.28	69.79	49.92	44.53	71.56	0	64.81
credit-g	76.02	50.03	191.3	76.19	49.83	164.2	76.23	52.94	174.7	75.89	39.74	150.2	75.81	49.79	179.6	75.95	0	299.9
Cleaveland	63.9	50.34	39.87	63.76	49.91	39.02	64.48	53.14	42.83	64.93	38.6	33.52	64.1	49.75	39.97	63.21	0	53.55
Ecoli	88.03	50.35	46.58	88.79	50.53	44.08	88.48	51.57	43.52	89.27	44.51	41.56	88.06	50.28	46.31	87.36	0	62.1
Glass	68.9	50.2	28.92	69.05	50.48	28.97	70.9	52.77	29.5	71.05	39.6	24.86	69.71	50.39	29.34	65.67	0	33.95
Hungarian	65.86	50.38	38.78	66.76	49.4	39.26	66.9	52.18	41.44	67.83	37.21	33.45	66.34	49.81	41.78	63.86	0	52.12
Iris	96.4	49.84	17.63	97.07	49.92	17.54	97.67	49.93	18.01	97.67	46.72	16.98	96.6	49.73	17.81	95.5	0	21.45
Liver	69.12	50.55	46.28	70.18	50.19	46.43	70.97	52.77	49.63	71.53	39.31	41.73	70.21	50.59	44.61	72.47	0	58.26
Pima Indians	76.88	50.02	116	76.57	50.38	138.2	77.71	53.21	130.4	78.37	38.67	102.4	77.22	49.68	111.6	76.92	0	126.7
Post Operative	71.63	56.58	11.57	71.63	56.37	11.14	72.25	56.58	11.85	71.5	56.37	12.06	72.13	56.65	11.59	71.25	0	11.87
Transfusion	78.97	50.23	112	79.55	49.71	107	79.51	51.68	107.3	79.18	38.16	84.76	79.51	50.3	95.94	78.61	0	135.2
vertebral-3c	85.45	50.21	36.24	87.42	50.07	37.22	86.97	51.82	37.88	87.65	42.32	34.97	86.77	49.90	37.76	85.61	0	53.31
Voting	96.21	50.13	53.17	95.88	49.73	54.23	96.53	51.7	55.04	96.53	43.87	49.15	96.47	50.23	58.05	95.77	0	83.07
Waveform	86.79	49.97	1563	86.77	49.99	1608	86.59	39.92	1132	86.77	39.83	1300	86.79	50.03	1597	86.98	0	2501
Wine	97.59	49.94	15.61	97.71	49.66	16.36	98.18	50.69	17.38	97.76	49.11	17.67	97.94	50.19	17.37	97.47	0	32.58
Yeast	60.2	50.02	216.1	61.04	207	50.1	61.49	52.98	223	60.92	40.73	185.5	60.91	50.02	209.5	59.45	0	306
Zoo	95.5	51.49	13.08	96.9	50.94	12.78	97.3	51.57	13.41	97	50.26	12.31	96.6	50.44	13.01	95	0	17.74
Average	**78.60**	**50.57**	**185.01**	**79.14**	**57.91**	**169.01**	**79.71**	**51.91**	**173.81**	**79.60**	**42.49**	**144.31**	**78.76**	**50.02**	**177.05**	**78.51**	**0**	**288.99**

Table 4.30: Wrapper-based proposed techniques vs ADR-Miner [17]

Dataset	**FFISA**		**FPISA**		**SSISA**		**CSISA**		**BISA**		**ADR-Miner [17]**	
	Accur	**Storg**	**Accur**	**Storg**	**Accur**	**Storg**	**Accur**	**Stor**	**Accur**	**Storg**	**Accur**	**Storg**
Breast T	68.30	51.09	69.90	50.88	71.30	53.26	69.50	47.41	67.60	50.66	60.64	23.98
Credit-g	76.02	50.03	76.19	49.83	76.23	52.94	75.89	39.74	75.81	49.79	74.1	19.31
Ecoli	88.03	50.35	88.79	50.53	88.48	51.57	89.27	44.51	88.06	50.28	81.34	21.33
Glass	68.90	50.2	69.05	50.48	70.90	52.77	71.05	39.60	69.71	50.39	69.64	31.4
Iris	96.40	49.84	97.07	49.92	97.67	49.93	97.67	46.72	96.60	49.73	92.56	42.08
Liver	69.12	50.55	70.18	50.19	70.97	52.77	71.53	39.31	70.21	50.59	58.56	17.55
Transfusion	78.97	50.23	79.55	49.71	79.51	51.68	79.18	38.16	79.51	50.3	72.31	21.88
Vertebral-3c	85.45	50.21	87.42	50.07	86.97	37.88	87.65	34.97	86.77	37.76	83.55	23.30
Voting	96.21	50.13	95.88	49.73	96.53	51.7	96.53	43.87	96.47	50.23	95.46	12
Zoo	95.50	51.49	96.90	50.94	97.30	51.57	97.00	50.26	96.60	50.44	98.75	52.78
Average	**82.29**	**50.41**	**83.09**	**50.23**	**83.59**	**50.61**	**83.53**	**42.46**	**82.73**	**49.02**	**78.69**	**26.56**

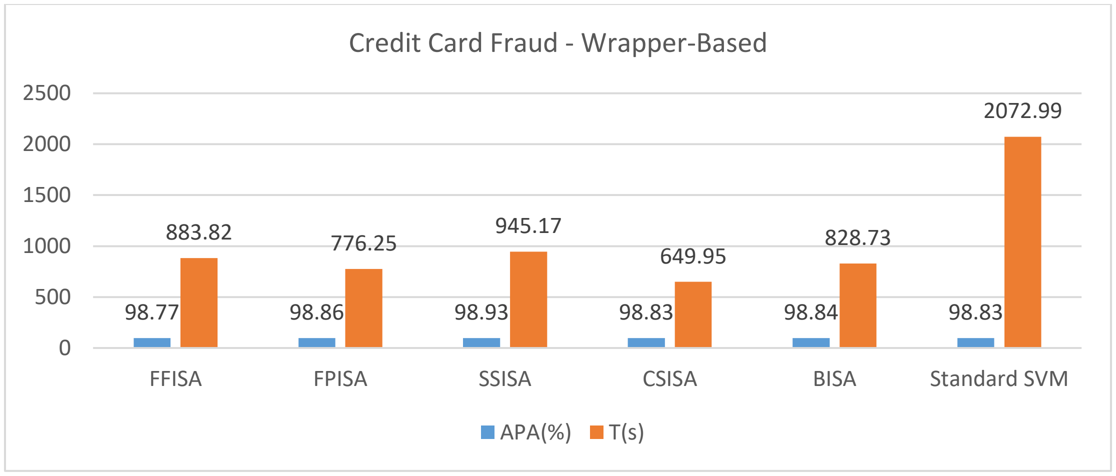

Figure 4.5: Wrapper-based techniques vs. standard SVM (credit card fraud)

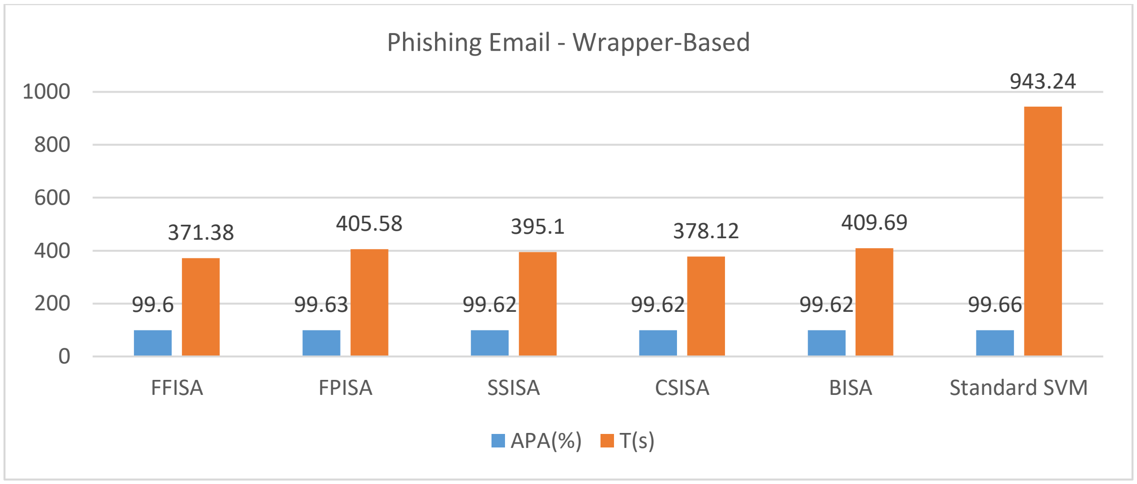

Figure 4.6: Wrapper-based techniques vs. standard SVM (phishing email)

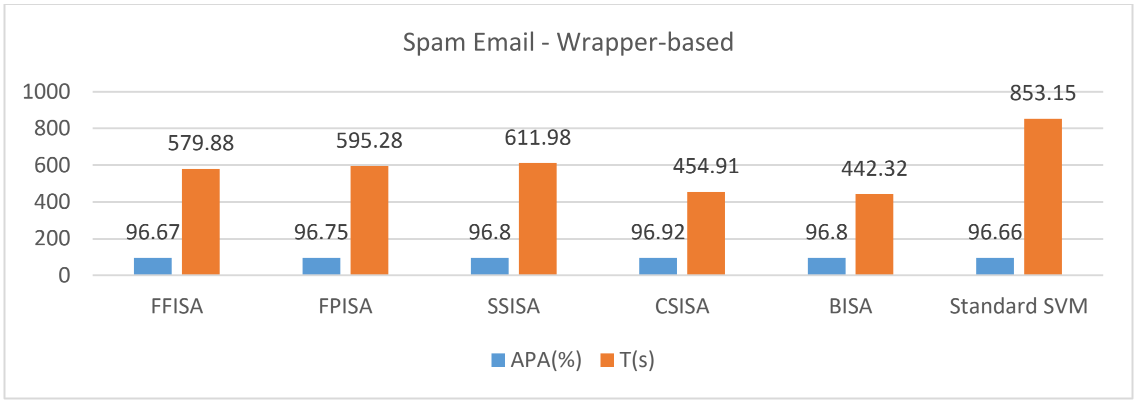

Figure 4.7: Wrapper-based techniques vs. standard SVM (spam email)

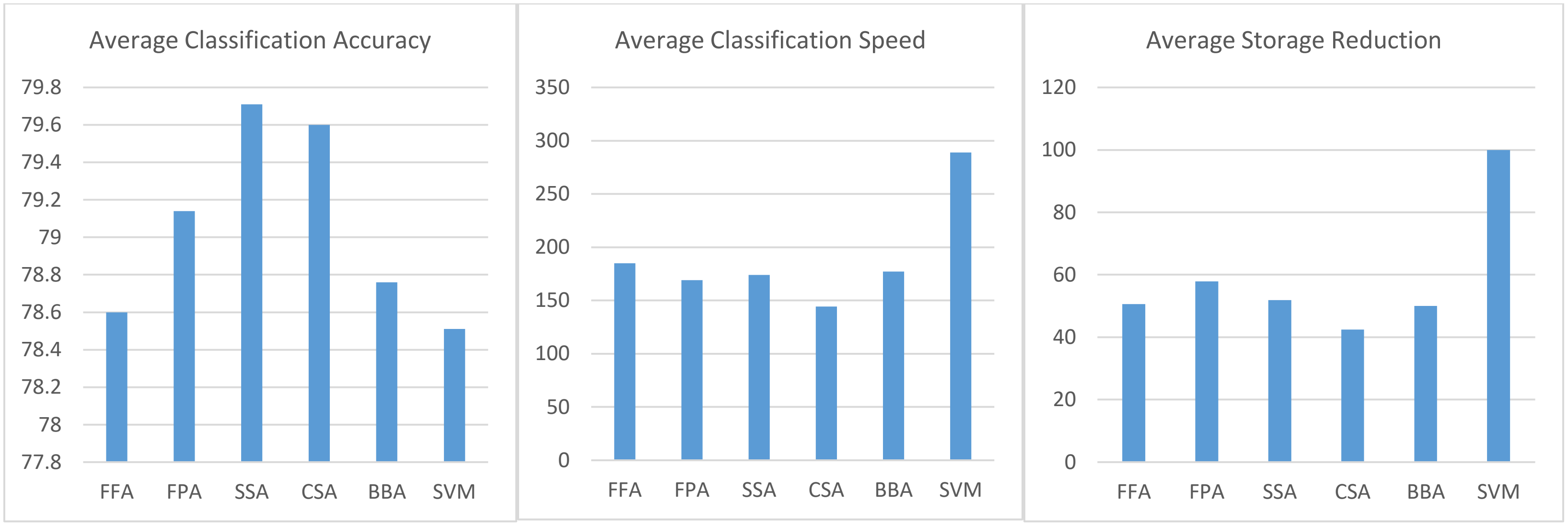

Figure 4.8: Wrapper-based techniques vs. standard svm (UCI datasets)

4.4.3 Statistical analysis

In this section, two different statistical test were conducted on the results obtained by the proposed filter-based and wrapper-based instance selection techniques. The tests are performed with the primary aim of showing that the proposed instance-selection techniques are statistically significantly faster than the standard SVM and two existing instance-selection techniques. Firstly, the Friedman's non-parametric test for multiple comparisons is used to check if there are any statistically significant differences between the proposed techniques, standard SVM, CLUS [1] and KNN [8]. Tables 4.31 – 4.37 report the mean rank, standard deviation, chi-square and p-value for the statistical analysis. As shown in the tables, for credit card fraud, the resulting Freidman statistics for all the filter-based techniques is 300. Taking note that the confidence level is 95%, the critical value in a chi-squared distribution with 3% degrees of freedom is 7.815. Since 300 is greater than 7.815, it can be concluded, with a 95% confidence level, that there are significant differences between the proposed filter-based techniques, standard SVM, CLUS [1] and KNN [8]. Similarly, as shown in Tables 4.31 – 4.37, the chi-square value for all the tests conducted on the phishing and spam email results is greater than 280. Since 280 is greater than 7.815, it can also be concluded with a 95% degree of confidence, that there are significant differences between the proposed filter-based techniques, standard SVM, CLUS [1] and KNN [8].

Table 4.31: Average rank from Friedman's non-parametric test for EDISA

Credit Card ($\chi^2 = 300$)			**Phishing Email ($\chi^2 = 288.25$)**			**Spam Email ($\chi^2 = 300$)**		
Algorithm	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**
SVM	4.00	16.114	SVM	4.00	2.901	SVM	4.00	3.052
CLUS	3.00	4.174	CLUS	2.89	9.135	CLUS	3.00	2.591
KNN	2.00	1.651	KNN	2.11	1.810	KNN	2.00	0.658
EDISA	1.00	0.295	EDISA	1.00	1.021	EDISA	1.00	0.487

Table 4.32: **Average rank from Friedman's non-parametric test for ACOISA**

Credit Card (χ^2 = 300)			**Phishing Email (χ^2 = 300)**			**Spam Email (χ^2 = 300)**		
Algorithm	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**
SVM	4.00	16.114	SVM	4.00	2.901	SVM	4.00	3.052
CLUS	3.00	4.174	CLUS	2.89	9.135	CLUS	3.00	2.591
KNN	2.00	1.651	KNN	2.11	1.810	KNN	2.00	0.658
ACOISA	1.00	0.510	ACOISA	1.00	2.044	ACOISA	1.00	0.621

Table 4.33: **Average rank from Friedman's non-parametric test for filter-based FFISA**

Credit Card Fraud (χ^2 = 300)			**Phishing Email (χ^2 = 288.25)**			**Spam Email (χ^2 = 300)**		
Algorithm	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**
SVM	4.00	16.114	SVM	4.00	2.901	SVM	4.00	3.052
CLUS	3.00	4.174	CLUS	2.89	9.135	CLUS	3.00	2.591
KNN	2.00	1.651	KNN	2.11	1.810	KNN	2.00	0.658
FFISA	1.00	0.964	FFISA	1.00	2.090	FFISA	1.00	0.393

Table 4.34: **Average rank from Friedman's non-parametric test for filter-based FPISA**

Credit Card Fraud (χ^2 = 300)			**Phishing Email (χ^2 = 288.25)**			**Spam Email (χ^2 = 300)**		
Algorithm	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**
SVM	4.00	16.114	SVM	4.00	2.901	SVM	4.00	3.052
CLUS	3.00	4.174	CLUS	2.89	9.135	CLUS	3.00	2.591
KNN	2.00	1.651	KNN	2.11	1.810	KNN	2.00	0.658
FPISA	1.00	0.580	FPISA	1.00	1.821	FPISA	1.00	0.548

Table 4.35: **Average rank from Friedman's non-parametric test for filter-based SSISA**

Credit Card Fraud (χ^2 = 300)			**Phishing Email (χ^2 = 283.76)**			**Spam Email (χ^2 = 300)**		
Algorithm	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**
SVM	4.00	16.114	SVM	4.00	2.901	SVM	4.00	3.052
CLUS	3.00	4.174	CLUS	2.87	9.135	CLUS	3.00	2.591
KNN	2.00	1.651	KNN	2.11	1.810	KNN	2.00	0.658
SSISA	1.00	0.757	SSISA	1.00	2.501	SSISA	1.00	0.861

Table 4.36: Average rank from Friedman's non-parametric test for filter-based CSISA

Credit Card Fraud ($\chi^2 = 300$)			Phishing Email ($\chi^2 = 288.25$)			Spam Email ($\chi^2 = 300$)		
Algorithm	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**
SVM	4.00	16.114	SVM	4.00	2.901	SVM	4.00	3.052
CLUS	3.00	4.174	CLUS	2.89	9.135	CLUS	3.00	2.591
KNN	2.00	1.651	KNN	2.11	1.810	KNN	2.00	0.658
CSISA	1.00	0.278	CSISA	1.00	1.364	CSISA	1.00	0.279

Table 4.37: Average rank from Friedman's non-parametric test for filter-based BISA

Credit Card Fraud ($\chi^2 = 300$)			Phishing Email ($\chi^2 = 288.25$)			Spam Email ($\chi^2 = 300$)		
Algorithm	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**
SVM	4.00	16.114	SVM	4.00	2.901	SVM	4.00	3.052
CLUS	3.00	4.174	CLUS	2.89	9.135	CLUS	3.00	2.591
KNN	2.00	1.651	KNN	2.11	1.810	KNN	2.00	0.658
BISA	1.00	0.920	BISA	1.00	1.943	BISA	1.00	0.410

Tables 4.38 – 4.42 show the average ranking, standard deviation and chi-square value of the tests conducted on the proposed wrapper-based techniques for credit card fraud, phishing emails and spam emails. As shown in the tables, the chi-square value for all the Freidman tests conducted on the three datasets is 100. Since 100 is greater than 7.815, it can also be concluded with a 95% confidence level that there are significant differences between the five proposed wrapper-based techniques and the standard SVM. Moreover, since the computed p-values for the three datasets are all < 0.0001, and the number of variables (or number of compared techniques) is two, it can be concluded, with a 95% confidence level, that the proposed wrapper-based techniques are significantly faster than standard SVM.

Table 4.38: Average rank from Friedman's non-parametric test for wrapper-based FFISA

Credit Card ($\chi^2 = 100$)			Phishing Email ($\chi^2 = 100$)			Spam Email ($\chi^2 = 100$)		
Algorithm	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**
SVM	2.00	16.114	SVM	2.00	2.901	SVM	2.00	3.052
FFISA	1.00	7.194	FFISA	1.00	7.217	EDISA	1.00	3.280

Table 4.39: Average rank from Friedman's non-parametric test for wrapper-based FPISA

Credit Card ($\chi^2 = 100$)			**Phishing Email** ($\chi^2 = 100$)			**Spam Email** ($\chi^2 = 100$)		
Algorithm	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**
SVM	2.00	16.114	SVM	2.00	2.901	SVM	2.00	3.052
FPISA	1.00	7.781	FPISA	1.00	6.430	FPISA	1.00	2.767

Table 4.40: Average rank from Friedman's non-parametric test for wrapper-based SSISA

Credit Card ($\chi^2 = 100$)			**Phishing Email** ($\chi^2 = 100$)			**Spam Email** ($\chi^2 = 100$)		
Algorithm	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**
SVM	2.00	16.114	SVM	2.00	2.901	SVM	2.00	3.052
SSISA	1.00	8.770	SSISA	1.00	7.175	SSISA	1.00	3.833

Table 4.41: Average rank from Friedman's non-parametric test for wrapper-based CSISA

Credit Card ($\chi^2 = 100$)			**Phishing Email** ($\chi^2 = 100$)			**Spam Email** ($\chi^2 = 100$)		
Algorithm	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**
SVM	2.00	16.114	SVM	2.00	2.901	SVM	2.00	3.052
CSISA	1.00	11.298	CSISA	1.00	5.966	CSISA	1.00	5.881

Table 4.42: Average rank from Friedman's non-parametric test for wrapper-based BISA

Credit Card ($\chi^2 = 100$)			**Phishing Email** ($\chi^2 = 100$)			**Spam Email** ($\chi^2 = 100$)		
Algorithm	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**
SVM	2.00	16.114	SVM	2.00	2.901	SVM	2.00	3.052
BISA	1.00	6.955	BISA	1.00	5.101	BISA	1.00	4.954

All of the conducted statistical tests showed that there are significant differences between the proposed filter-based techniques and the compared techniques. However, the tests did not specify the statistical difference between each technique. Hence, to obtain the statistical difference, Holm's post-hoc test was conducted, using each of the proposed filter-based techniques as control algorithms and the compared techniques as independent algorithms. The adjusted and unadjusted p-values obtained from the post-hoc tests for all the proposed techniques are < 0.0001. This is because the significant differences between the proposed filter-based techniques and the compared techniques are all very large ($p < 0001$). Since 0.0001 is less than 0.05, it can be concluded with a 95% confidence level, that the proposed filter-based instance selection techniques are significantly faster than the standard SVM. Holm's post-hoc test was not conducted for the wrapper-based techniques because the wrapper-based techniques are compared to one variable (i.e. the standard SVM), and the post-hoc test can only be conducted on results with more than two variables. The wrapper-based techniques are not compared to CLUS [1] and KNN [8] because both techniques are filter-based techniques, hence the comparison would not be a fair assessment.

Finally, to identify the best filter-based and wrapper-based technique, Friedman's test was conducted on the results produced by the proposed techniques. Firstly, Friedman's test was conducted on the seven filter-based techniques, then Friedman's test was also conducted on the five wrapper-based techniques. Table 4.43 shows the Freidman's test result for the filter-based techniques. As shown in the table, the resulting Freidman's statistics for phishing email is 199.40. Moreover, the critical value at 95% confidence level and 6% degree of freedom is 12.592. Since 199.40 is greater than 12.592, for phishing email, it can be concluded that there are significant differences among the results produced by the seven proposed filter-based techniques. Furthermore, as shown in Table 4.43, the resulting Freidman's statistics for spam emails and credit card fraud is 503.24 and 548.99, respectively. Since both values are greater than 12.592, for spam emails and credit card fraud, it can be concluded that there are significant differences among the results produced by the seven proposed filter-based techniques. Moreover, as shown in Table 4.43, EDISA achieved the best rank for phishing emails, and CSISA achieved the best rank for spam emails and credit card fraud. Furthermore, Tables 4.44 shows the Freidman's test result for the wrapper-based techniques. As shown in the table, the chi-square values for credit card fraud, phishing emails and spam emails is 300, 63.11 and 328.64, respectively. Moreover, the critical value at a 95% confidence level and a 4% degree of freedom is 9.488. Since the three chi-square values are greater than 9.488, it can be concluded that there are significant differences among the results produced by the five wrapper-based techniques. Moreover, as shown in Table 4.44, FFISA achieved the best rank for phishing emails, BISA achieved the best rank for spam emails and CSISA achieved the best rank for credit card fraud.

To select the best performing algorithm among the proposed techniques, Holm's post-hoc test for multiple comparisons was conducted on the results produced by each of the techniques. Initially, Holm's test was conducted on the filter-based techniques, then Holm's test was conducted on the wrapper-based techniques. Tables 4.45 – 4.50 show the mean, standard deviation and significance level for the Holm's tests. The values underlined indicate that there is a significant difference between the underlined algorithm and the compared algorithm. As shown in Table 4.43, for phishing emails, EDISA produced the best rank, followed by CSISA. Moreover, as shown in Table 4.47, there is a significant difference between EDISA and the following algorithms: ACOISA, FFISA, FPISA, SSISA, CSISA and BISA. Therefore, it can be concluded with a 95% confidence level that EDISA is significantly faster than the other filter-based techniques, when applied to phishing emails. Furthermore, as shown in Table 4.47, there is a significant difference between CSISA and ACOISA, FFISA, FPISA, SSISA and BISA, hence it can be concluded with a 95% confidence level that CSISA is significantly faster than ACOISA, FFISA, FPISA, SSISA and BISA, when applied to spam emails. Overall, it can be concluded that EDISA and CSISA, respectively, achieved the first and second best training speeds for phishing emails.

As shown in Table 4.43, for spam emails, CSISA achieved the best rank, followed by BISA. Also, as shown in Table 4.46, there is a significant difference between CSISA and EDISA, ACOISA, FFISA, FPISA, SSISA and BISA. Hence, for spam emails, it can be concluded with a 95% confidence level that CSISA is significantly better than the other filter-based techniques. As shown in Table 4.43, BISA produced the second best rank. Also, Table 4.46 indicates that there is a significant difference between BISA and EDISA, ACOISA, FFISA, FPISA and SSISA. Hence, for spam emails, it can be concluded with a 95% confidence level that BISA is significantly better than EDISA, ACOISA, FFISA, FPISA and SSISA. Table 4.43 shows the Friedman's test result for credit card fraud, and as shown in the table, CSISA produced the best rank, followed by SSISA. Also, as shown in Table 4.47, there is a significant difference between CSISA and the other six filter-based techniques. Moreover, as shown in Table 4.47, there is a significant difference between SSISA and EDISA, ACOISA, FFISA and FPISA. Therefore, for credit card fraud it can be concluded with a 95% confidence level, that CSISA achieved the best training speed, followed by SSISA.

Table 4.44 shows the average rank, chi-square value, *p*-value and standard deviation for the Freidman's test conducted on the wrapper-based techniques. As shown in the table, for credit card fraud, CSISA produced the best rank, followed by BISA. Moreover, for spam email, BISA produced the best rank, followed by CSISA. Also, for phishing email, FFISA produced the best rank, followed by CSISA. Furthermore, as shown in Table 4.50, there is a significant difference between CSISA and the other wrapper-based techniques. Also, there is a significant difference between BISA and FFISA, FPISA and SSISA. Therefore, it can be concluded, with a 95% confidence level that CSISA and FPISA produced the first and second best results for credit card fraud, when compared to the other wrapper-based techniques. Moreover, as shown in Table 4.46, there is a significant difference between BISA and FFISA, FPISA and SSISA. Table 4.46 also shows that there is a significant difference between CSISA and FFISA, FPISA and SSISA. Hence, it can be concluded with 95% confidence level, that BISA and CSISA achieved the first and second best result for spam email. Furthermore, as shown in Table 4.48, there is a significant difference between FFISA and FPISA, BISA. Moreover, Table 4.46 also shows that there is a significant difference between CSISA and FPISA. Hence, it can be concluded with 95% confidence level, that FFISA and CSISA produced the first and second best training speeds for phishing emails.

Table 4.43: Friedman's non-parametric test results for filter-based techniques

Algorithm	**Credit Card** ($\chi^2 = 548.996$)		**Phishing Email** ($\chi^2 = 199.404$)		**Spam Email** ($\chi^2 = 503.236$)	
($\boldsymbol{p} < 0.001$)	**Ranking**	**S.Dev**	**Ranking**	**S.Dev**	**Ranking**	**S.Dev**
EDISA	2.85	0.295	1.86	1.021	3.73	0.487
ACOISA	6.44	0.510	4.42	4.476	4.67	0.621
FFISA	5.41	0.964	4.43	4.595	4.31	0.393
FPISA	3.96	0.581	4.72	4.882	5.50	0.548
SSISA	2.26	0.757	5.47	5.327	6.79	0.861
CSISA	1.00	0.278	2.75	3.054	1.07	0.279
BISA	6.08	0.920	4.36	4.562	1.94	0.409

Table 4.44: Friedman's non-parametric test results for wrapper-based techniques

Algorithm	**Credit Card** ($\chi^2 = 300$)		**Phishing Email** ($\chi^2 = 63.112$)		**Spam Email** ($\chi^2 = 328.64$)	
($\boldsymbol{p} < 0.001$)	**Ranking**	**S.Dev**	**Ranking**	**S.Dev**	**Ranking**	**S.Dev**
FFISA	4.06	7.194	2.26	7.217	3.45	3.280
FPISA	2.20	7.781	3.74	6.430	3.93	2.767
SSISA	4.62	8.770	2.96	7.175	4.62	3.833
CSISA	1.24	11.22	2.53	5.966	1.61	5.881
BISA	2.88	6.955	3.51	5.101	1.39	4.954

Table 4.45: Holm’s post hoc test for filter-based techniques on phishing email

Compared Algorithm (i)	Algorithm (j)	Mean Difference (i - j)	Significance Level
EDISA	ACOISA	-2.6756	0.000
	FFISA	-2.7952	0.000
	FPISA	-3.0816	0.000
	SSISA	-3.5273	0.000
	CSISA	-1.2543	0.000
	BISA	-2.7616	0.000
ACOISA	EDISA	2.6756	0.000
	FFISA	-0.1196	1.000
	FPISA	-0.4061	1.000
	SSISA	-0.8518	0.030
	CSISA	1.4212	0.000
	BISA	-0.086	1.000
FFISA	EDISA	2.7952	0.000
	ACOISA	0.1196	1.000
	FPISA	-0.2864	1.000
	SSISA	-0.7322	0.128
	CSISA	1.5409	0.000
	BISA	0.0336	1.000
FPISA	EDISA	3.0816	0.000
	ACOISA	0.4061	1.000
	FFISA	0.2864	1.000
	SSISA	-0.4457	1.000
	CSISA	1.8273	0.000
	BISA	0.3201	1.000
SSISA	EDISA	3.5273	0.000
	ACOISA	0.8518	0.030
	FFISA	0.7322	0.128
	FPISA	0.4457	1.000
	CSISA	2.273	0.000
	BISA	0.7658	0.087

CSISA	EDISA	1.2543	0.000
	ACOISA	-1.4212	0.000
	FFISA	-1.5409	0.000
	FPISA	-1.8273	0.000
	SSISA	-2.273	0.000
	BISA	-1.5072	0.000
BISA	EDISA	2.7616	0.000
	ACOISA	0.086	1.000
	FFISA	-0.0336	1.000
	FPISA	-0.3201	1.000
	SSISA	-0.7658	0.087
	CSISA	1.5072	0.000

Table 4.46: **Holm's post hoc test for filter-based techniques on spam email**

Compared Algorithm (i)	Algorithm (j)	Mean Difference (i - j)	Significance Level
EDISA	ACOISA	-0.348	0.000
	FFISA	-0.1666	0.638
	FPISA	-0.5976	0.000
	SSISA	-1.4309	0.000
	CSISA	1.6966	0.000
	BISA	1.0566	0.000
ACOISA	EDISA	.3480	0.000
	FFISA	0.1814	0.388
	FPISA	-0.2496	0.025
	SSISA	-1.0829	0.000
	CSISA	2.0446	0.000
	BISA	1.4045	0.000
FFISA	EDISA	0.1666	0.638
	ACOISA	-0.1814	0.388
	FPISA	-0.4310	0.000
	SSISA	-1.2642	0.000
	CSISA	1.8633	0.000
	BISA	1.2232	0.000
FPISA	EDISA	0.5976	0.000
	ACOISA	0.2496	0.025
	FFISA	0.4310	0.000
	SSISA	-0.8333	0.000
	CSISA	2.2943	0.000
	BISA	1.6542	0.000
SSISA	EDISA	1.4309	0.000
	ACOISA	1.0829	0.000
	FFISA	1.2642	0.000
	FPISA	0.8333	0.000
	CSISA	3.1275	0.000
	BISA	2.4874	0.000
CSISA	EDISA	-1.6966	0.000
	ACOISA	-2.0446	0.000

	FFISA	<u>-1.8633</u>	0.000
	FPISA	<u>-2.2943</u>	0.000
	SSISA	<u>-3.1275</u>	0.000
	BISA	<u>-0.6401</u>	0.000
BISA	EDISA	<u>-1.0566</u>	0.000
	ACOISA	<u>-1.4045</u>	0.000
	FFISA	<u>-1.2232</u>	0.000
	FPISA	<u>-1.6542</u>	0.000
	SSISA	<u>-2.4874</u>	0.000
	CSISA	<u>0.6401</u>	0.000

Table 4.47: Holm’s post hoc test for filter-based techniques on credit card email

Compared Algorithm (i)	Algorithm (j)	Mean Difference (i - j)	Significance Level
EDISA	ACOISA	-2.8872	0.000
	FFISA	-2.2547	0.000
	FPISA	-0.7640	0.000
	SSISA	0.5982	0.000
	CSISA	2.5253	0.000
	BISA	-2.4911	0.000
ACOISA	EDISA	2.8872	0.000
	FFISA	0.6325	0.000
	FPISA	2.1232	0.000
	SSISA	3.4854	0.000
	CSISA	5.4125	0.000
	BISA	0.3961	0.001
FFISA	EDISA	2.2547	0.000
	ACOISA	-0.6325	0.000
	FPISA	1.4907	0.000
	SSISA	2.8529	0.000
	CSISA	4.7799	0.000
	BISA	-0.2364	0.260
FPISA	EDISA	0.7640	0.000
	ACOISA	-2.1232	0.000
	FFISA	-1.4907	0.000
	SSISA	1.3622	0.000
	CSISA	3.2893	0.000
	BISA	-1.7271	0.000
SSISA	EDISA	-0.5982	0.000
	ACOISA	-3.4854	0.000
	FFISA	-2.8529	0.000
	FPISA	-1.3622	0.000
	CSISA	1.9271	0.000
	BISA	-3.0893	0.000
CSISA	EDISA	-2.5253	0.000
	ACOISA	-5.4125	0.000

	FFISA	<u>-4.7799</u>	0.000
	FPISA	<u>-3.2893</u>	0.000
	SSISA	<u>-1.9271</u>	0.000
	BISA	<u>-5.0164</u>	0.000
BISA	EDISA	<u>2.4911</u>	0.000
	ACOISA	<u>-0.3961</u>	0.001
	FFISA	0.2364	0.260
	FPISA	<u>1.7271</u>	0.000
	SSISA	<u>3.0893</u>	0.000
	CSISA	<u>5.0164</u>	0.000

Table 4.48: Holm's post hoc test for wrapper-based techniques on phishing email

Compared Algorithm (*i*)	**Algorithm (*j*)**	**Mean Difference (*i* - *j*)**	**Significance Level**
FFISA	FPISA	<u>-3.4203</u>	0.002
	SSISA	-2.3716	0.094
	CSISA	-0.6736	1.000
	BISA	<u>-3.8308</u>	0.000
FPISA	FFISA	<u>3.4203</u>	0.002
	SSISA	1.0487	1.000
	CSISA	<u>2.7467</u>	0.026
	BISA	-0.4105	1.000
SSISA	FFISA	2.3716	0.094
	FPISA	-1.0487	1.000
	CSISA	1.6980	0.623
	BISA	-1.4592	1.000
CSISA	FFISA	0.6736	1.000
	FPISA	<u>-2.7467</u>	0.026
	SSISA	-1.6980	0.623
	BISA	<u>-3.1572</u>	0.006
BISA	FFISA	<u>3.8308</u>	0.000
	FPISA	0.4105	1.000
	SSISA	1.4592	1.000
	CSISA	<u>3.1572</u>	0.006

Table 4.49: Holm's post hoc test for wrapper-based techniques on spam email

Compared Algorithm (*i*)	Algorithm (*j*)	Mean Difference (*i* - *j*)	Significance Level
FFISA	FPISA	-1.5403	0.115
	SSISA	-3.2102	0.000
	CSISA	12.4965	0.000
	BISA	13.7551	0.000
FPISA	FFISA	1.5403	0.115
	SSISA	-1.6699	0.062
	CSISA	14.0368	0.000
	BISA	15.2954	0.000
SSISA	FFISA	3.2102	0.000
	FPISA	1.6699	0.062
	CSISA	15.7068	0.000
	BISA	16.9653	0.000
CSISA	FFISA	-12.4965	0.000
	FPISA	-14.0368	0.000
	SSISA	-15.7068	0.000
	BISA	1.2586	0.388
BISA	FFISA	-13.7551	0.000
	FPISA	-15.2954	0.000
	SSISA	-16.9653	0.000
	CSISA	-1.2586	0.388

Table 4.50: Holm's post hoc test for wrapper-based techniques on credit card fraud

Compared Algorithm (i)	Algorithm (j)	Mean Difference (i - j)	Significance Level
FFISA	FPISA	10.7571	0.000
	SSISA	-6.1349	0.000
	CSISA	23.3865	0.000
	BISA	5.5086	0.000
FPISA	FFISA	-10.7571	0.000
	SSISA	-16.8920	0.000
	CSISA	12.6294	0.000
	BISA	-5.2485	0.000
SSISA	FFISA	6.1349	0.000
	FPISA	16.8920	0.000
	CSISA	29.5214	0.000
	BISA	11.6435	0.000
CSISA	FFISA	-23.3865	0.000
	FPISA	-12.6294	0.000
	SSISA	-29.5214	0.000
	BISA	-17.8779	0.000
BISA	FFISA	-5.5086	0.000
	FPISA	5.2485	0.000
	SSISA	-11.6435	0.000
	CSISA	17.8779	0.000

Overall, based on all of the statistical test results, filter-based CSISA achieved the best results for spam emails and credit cards, and also achieved the second best result for phishing emails. In addition, the wrapper-based CSISA achieved the best results for credit card fraud, and the second best result for spam and phishing emails. Therefore, it can be concluded that CSISA produced the best training speed, compared to the other proposed filter-based and wrapper-based techniques.

4.5 Chapter summary

This chapter presented the experimental results for the filter and wrapper based techniques proposed in this thesis. As shown in Figures 4.1 – 4.3, the proposed filter-based techniques improved SVM classification speed by over 93%, without significantly affecting SVM classification accuracy. Moreover, Tables 4.29 – 4.31 and Figures 4.5 – 4.8 show the comparisons between the proposed wrapper-based techniques and standard SVM. For each table, the best three examples of predictive accuracy are underlined. As shown in each table, for both credit card fraud and spam emails, the proposed wrapper-based technique improved SVM predictive accuracy and also reduced the training dataset size by an average of 50%. This indicates that the proposed wrapper-based techniques require approximately half (i.e. 50%) of the training dataset to produce improved classification models. Although the primary objective of the proposed wrapper-based techniques is to improve SVM predictive accuracy, the techniques also simultaneously improved SVM classification speed by over 32% for spam emails, 56% for phishing emails and 54% for credit card fraud. Generally, the results obtained by the proposed filter-based and wrapper-based techniques show that they are fast and accurate e-fraud detection and instance-selection techniques. The proposed techniques will be highly suited to applications that process large datasets with limited storage space.

Chapter 5
Summary, Conclusion and Future Research Directions

In many domains today, such as data mining, engineering and science, ML-based solutions are highly essential [29]. This thesis proposes intelligent instance-selection techniques for improving SVM classification speed and predictive accuracy. Many experiments are performed to validate the performance of the techniques and the experimental results show that the proposed techniques achieved better results, compared to the standard SVM and some existing instance-selection techniques. This section summarizes and concludes this thesis, and also provide directions for future research.

5.1 Summary

As shown in the results, SVM performs slower when applied to large datasets. Its training time is $O(n^2)$, where n is the number of training instances [257, 258]. This implies that n plays a significant role in SVM speed. Hence, SVM speed and computational complexity can be significantly improved by reducing n (i.e. number of instances). As shown in Section 2.1, many SVM speed optimization approaches have been used in literature. Some studies introduced feature selection techniques, while others introduced parameter optimization and instance selection techniques. Among the three class of techniques, instance selection techniques produced one of the best results [6]. Moreover, as shown in the comprehensive survey presented in Chapter two, most of the existing instance selection techniques focused on clustering. Very few studies explored NI algorithms. Therefore, this thesis propose seven filter-based and five wrapper-based intelligent techniques for improving SVM training speed and predictive accuracy. The proposed techniques can be applied to different data mining problems, however, in this research, the proposed techniques are applied to e-fraud detection problems, with a particular focus on three popular e-fraud types: credit card fraud, spam email and phishing email. Two set of experiments was performed to test the efficacy of the proposed techniques. The first set of experiments was performed to test the efficacy of the proposed filter-based techniques and the second set of experiments was performed to test the efficacy of the proposed wrapper-based techniques. In addition, the techniques was compared to standard SVM and some existing instance selection techniques. Experimental result reveal that the proposed techniques significantly improved SVM training speed and predictive accuracy. Two different statistical test was performed to further validate the efficacy of the proposed techniques and the analysis shows that the proposed techniques significantly improved SVM training speed in all cases. Statistical analysis was also performed to identify the best wrapper-based and filter-based techniques, and as shown in Section 4.4.3, CSISA outperform all the proposed techniques. Summarily, as shown in all the results presented in Chapter three, the proposed techniques are excellent SVM speed optimizers and ML-based e-fraud detection techniques. The filter-based techniques are suitable for applications that requires real time online training, while the wrapper-based techniques are suitable for applications that are very sensitive to slight drop in predictive accuracy, such as email classifiers.

5.2 Conclusion

In recent times, the volume of data produced by different sources worldwide is enormously increasing [198]. However, data can be useless if significant information cannot be extracted from them. Moreover, extracting information from huge datasets is very challenging. Hence, there is an obvious need for efficient information extraction tools. ML-based solutions are suitable tools for information extraction. They can be used to effectively extract relevant information from datasets. However, ML algorithms generally perform poorly when applied to large datasets [257, 258] . This is because large datasets typically contain many superfluous and harmful instances, which pose problems to the generalization performance of ML algorithms [6]. Hence, this thesis proposes intelligent optimization techniques for improving the speed of ML algorithms, with a particular focus on SVM. SVM is a well-known ML algorithm that is widely used to handle many real-world applications with great success. However, similar to other ML algorithms, SVM computational complexity deteriorates significantly when applied to massive datasets, thus making it unfitting for real-time applications.

As stated in the second and third objectives of this thesis (outlined in Section 1.3), this PhD research proposes seven intelligent filter-based and five wrapper-based instance-selection techniques for improving SVM predictive accuracy, training speed, generalization performance and computational complexity. The proposed techniques are inspired by FPA, FFA, SSA, BA and CSA. Additionally, two of the proposed techniques are boundary detection algorithms, inspired by edge detection in image processing and edge selection in ACO. The proposed techniques can be applied to different problems, however, as stated in the fourth objective of this research, the proposed techniques are used to handle three popular e-fraud detection problems: credit card fraud, phishing email detection and spam email detection. In addition, the proposed techniques are also validated on 21 other classification problems provided by the UCI dataset repository.

Different sets of experiments are performed to validate the efficiency of the proposed techniques, and experimental results show that the filter-based techniques excellently improved SVM training speed in 100% of the datasets used for evaluation, without significantly affecting SVM classification quality. Also, the results show that the wrapper-based techniques improved SVM predictive accuracy in 78% of the datasets (18 out of 23) used for evaluation, and simultaneously improved SVM training speed for all cases. Additionally, the experimental results show that the proposed techniques produced excellent storage reduction and speed-accuracy trade-off. Moreover, the results show that the proposed techniques are good ML-based e-fraud detection techniques. All of these clearly address the research questions outlined in Section 1.2. Furthermore, as stated in the sixth objective, two different statistical tests were conducted on the results produced by the proposed techniques. Firstly, Friedman's test was conducted, followed by Holm's test. As shown in Tables 4.31 – 4.42, it can be concluded with a 95% confidence level, that the proposed techniques are significantly faster than standard SVM and some existing instance-selection techniques. Statistical analysis was performed to identify the best technique among the proposed filter-based and wrapper-based techniques. As shown in Tables 4.43 – 4.50, the statistical results show that CSISA significantly outperforms the other proposed filter-based and wrapper-based techniques, in terms of training speed. This implies that CSISA is the fastest instance selection technique proposed in this thesis. Conclusively, as shown in all the results, the proposed techniques outperform standard SVM in both training speed and predictive accuracy, hence, they can be used in combination with standard SVM to produce better and faster classification models.

5.3 Future research directions

Big Data analytics is becoming an important tool in decision making for businesses and organizations. The rate of data growth is very alarming, and it is already going beyond the Exabyte limit. Moreover, the ability to make strategic and meaningful decisions depends on the reliability of data. Hence, there is an urgent need for fast and accurate tools for Big Data analytics. ML-based solutions are very useful and reliable data analytic tools. ML algorithms are known for their robustness [25], dynamic problem solving [25], accurate data mining and classification proficiency [25, 26]. However, contemporary mathematical models for ML algorithms are complex [205]. Further research could explore the development of simple hybrid ML-based algorithms that are very fast and highly accurate.

To improve the speed for classification or instance selection, variants of PSO and GA with the proposed NI-based techniques could be considered for better result [259]. In addition, the techniques designed in this research can be applied to other ML algorithms like ANN, RF, regression, etc.

In this work, data anonymization has not been considered. Riyazuddin and Balaram [260] propose 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.

Furthermore, the methods considered in this research are iterative in structure, future research can therefore explore the possible implementation of non-iterative approaches. In addition, alternative performance improvement strategies for ML algorithms could be to explore different methods for imbalanced datasets and deep learning.

References

[1] 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.

[2] M. N. Arbatskaya, K. Mukhopadhaya, and E. B. Rasmusen, "The Parking Lot Problem (December 19, 2006)," *Available at: https://ssrn.com/abstract=571101 or http://dx.doi.org/10.2139/ssrn.571101*, 2006.

[3] 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.

[4] 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.

[5] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations," *The Journal of Machine Learning Research*, vol. 2, pp. 243-264, 2001.

[6] 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.

[7] 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.

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

[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, 2006.

[10] H. Liu and H. Motoda, "On issues of instance selection," *Data Mining and Knowledge Discovery*, vol. 6, no. 2, pp. 115-130, 2002.

[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, 2001.

[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, 2001.

[13] J. A. Olvera-López, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "Sequential search for decremental edition," in *International Conference on Intelligent Data Engineering and Automated Learning*, 2005, pp. 280-285.

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

[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, 2005.

[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, 2008.

[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.

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

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

[20] A. Munoz, "Machine Learning and Optimization," *URL: https://www.cims.nyu.edu/~munoz/files/ml_optimization.pdf (accessed 30-Februrary-2017) [WebCite Cache ID 6fiLfZvnG]*, 2014.

[21] J. M. Buhmann. (2015), "Machine Learning". available at: https://ml2.inf.ethz.ch/courses/ml/ (accessed 08-December-2015).

[22] A. Bergholz, J. H. Chang, G. Paaß, F. Reichartz, and S. Strobel, "Improved Phishing Detection using Model-Based Features," *Proceedings of the Conference on Email and Anti-Spam (CEAS),* 2008.

[23] CollabraSpace. (2013), "Machine Learning for Cybersecurity". available at: http://www.collabraspace.com/news/machine-learning-for-cybersecurity/ (accessed 25-February-2015).

[24] T. O. Ayodele, *Types of Machine Learning Algorithms, New Advances in Machine Learning*. Yagang Zhang (Ed.): InTech, 2010, DOI: 10.5772/9385. Available at: https://www.intechopen.com/books/new-advances-in-machine-learning/types-of-machine-learning-algorithms.

[25] M. Behdad, L. Barone, M. Bennamoun, and T. French, "Nature-Inspired Techniques in the Context of Fraud Detection," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 42, no. 6, pp. 1273-1290, November, 2012.

[26] P. J. Fleming and R. C. Purshouse, "Evolutionary algorithms in control systems engineering: a survey," *Control engineering practice*, vol. 10, no. 11, pp. 1223-1241, 2002.

[27] 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.

[28] J. Fehr, K. Z. Arreola, and H. Burkhardt, "Fast Support Vector Machine Classification of Very Large Datasets," in Data Analysis, Machine Learning and Applications: Proceedings of the 31st Annual Conference of the Gesellschaft für Klassifikation e.V., Albert-Ludwigs-Universität Freiburg, March 7–9, 2007, C. Preisach, H. Burkhardt, L. Schmidt-Thieme, and R. Decker, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 11-18.

[29] S. Ertekin, "Learning in extreme conditions: Online and active learning with massive, imbalanced and noisy data," Doctor of Philosophy, Computer Science and Engineering, The Pennsylvania State University, 2009.

[30] 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).

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

[32] Andrea. (2016), "Credit Card Fraud Detection". available at: https://www.kaggle.com/dalpozz/creditcardfraud (accessed 27-October-2016).

[33] J. Nazario. (2005), "Phishing Corpus". available at: http://monkey.org/jose/wiki/doku.php?id=PhishingCorpus (accessed 27-April-2015).

[34] 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, pp. 1-27, 2011.

[35] M. Johnson. (2009), "SVM.NET". available at: http://www.matthewajohnson.org/software/svm.html (accessed 05-August-2014).

[36] P. Gaspar, J. Carbonell, and J. L. Oliveira, "On the parameter optimization of Support Vector Machines for binary classification," *Journal of Integrative Bioinformatics,* vol. 9, no. 3, pp. 33-43, 2012.

[37] M. S. Uzer, N. Yilmaz, and O. Inan, "Feature selection method based on artificial bee colony algorithm and support vector machines for medical datasets classification," *The Scientific World Journal,* vol. 2013, Article ID 419187, 10 Pages, 2013.

[38] M. A. Laamari and N. Kamel, "A hybrid bat based feature selection approach for intrusion detection," in Bio-Inspired Computing-Theories and Applications, ed: Springer, 2014, pp. 230-238.

[39] R. R. Rajalaxmi, "A Hybrid Binary Cuckoo Search and Genetic Algorithm for Feature Selection in Type-2 Diabetes," *Current Bioinformatics,* vol. 11, no. 5, pp. 490 - 499, 2016.

[40] D. Rodrigues, L. A. Pereira, R. Y. Nakamura, K. A. 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, 2014.

[41] 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.

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

[43] 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, 2014.

[44] B. Schölkopf, P. Simard, V. Vapnik, and A. Smola, "Improving the accuracy and speed of support vector machines," *Advances in neural information processing systems,* vol. 9, pp. 375-381, 1997.

[45] J. Guo, N. Takahashi, and T. Nishi, "A learning algorithm for improving the classification speed of support vector machines," in *Proceedings of the 2005 European Conference on Circuit Theory and Design, 2005.*, Cork, Ireland, Ireland, 2005, pp. 381-384.

[46] Y.-J. Lee and O. L. Mangasarian, "RSVM: Reduced support vector machines," in *Proceedings of the 2001 SIAM International Conference on Data Mining*, 2001, pp. 325-361.

[47] H. Lei and V. Govindaraju, "Speeding up multi-class SVM evaluation by PCA and feature selection," in *Proceedings of the Workshop on Feature Selection for Data Mining:*

Interfacing Machine Learning and Statistics, Newport Beach, CA, 2005, pp. 72-79.

[48] S. Garcı, I. Triguero, C. J. Carmona, and F. Herrera, "Evolutionary-based selection of generalized instances for imbalanced classification," *Knowledge-Based Systems*, vol. 25, no. 1, pp. 3-12, 2012.

[49] J. R. Cano, F. Herrera, and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 6, pp. 561-575, 2003.

[50] H. Shin and S. Cho, "Neighborhood property–based pattern selection for support vector machines," *Neural Computation*, vol. 19, no. 3, pp. 816-855, 2007.

[51] F. Angiulli and A. Astorino, "Scaling up support vector machines using nearest neighbor condensation," *IEEE Transactions on Neural Networks*, vol. 21, no. 2, pp. 351-357, 2010.

[52] F. Angiulli, "Fast nearest neighbor condensation for large data sets classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 11, pp. 1450-1464, 2007.

[53] C.-F. Tsai and K.-C. Cheng, "Simple instance selection for bankruptcy prediction," *Knowledge-Based Systems*, vol. 27, pp. 333-342, 2012.

[54] R. Koggalage and S. Halgamuge, "Reducing the number of training samples for fast support vector machine classification," *Neural Information Processing-Letters and Reviews*, vol. 2, no. 3, pp. 57-65, 2004.

[55] K. Z. Arreola, J. Fehr, and H. Burkhardt, "Fast support vector machine classification using linear SVMs," in *18th International Conference on Pattern Recognition (ICPR'06)*, 2006, pp. 366-369.

[56] F. Temitayo, O. Stephen, and A. Abimbola, "Hybrid GA-SVM for efficient feature selection in e-mail classification," *Computer Engineering and Intelligent Systems*, vol. 3, no. 3, pp. 17-28, 2012.

[57] D. R. Pereira, M. A. Pazoti, L. A. M. Pereira, and J. P. Papa, "A social-spider optimization approach for support vector machines parameters tuning," in *2014 IEEE Symposium on Swarm Intelligence*, Orlando, FL, USA, 2014, pp. 1-6.

[58] D. R. Pereira, M. A. Pazoti, L. A. Pereira, D. Rodrigues, C. O. Ramos, A. N. Souza, *et al.*, "Social-Spider Optimization-based Support Vector Machines applied for energy theft detection," *Computers & Electrical Engineering,* vol. 49, pp. 25-38, 2016.

[59] O. Hegazy, O. S. Soliman, and M. A. Salam, "Comparative Study between FPA, BA, MCS, ABC, and PSO Algorithms in Training and Optimizing of LS-SVM for Stock Market Prediction," *International Journal of Advanced Computer Research,* vol. 5, no. 18, p. 35, 2015.

[60] M. Boardman and T. Trappenberg, " A Heuristic for Free Parameter Optimization with Support Vector Machines," in *Proceedings of the 2006 IEEE International Joint Conference on Neural Networks (IJCNN 2006)*, 2006, pp. 1337-1344.

[61] F. Friedrichs and C. Igel, "Evolutionary tuning of multiple SVM parameters," *Neurocomputing,* vol. 64, pp. 107-117, 2005.

[62] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies–A comprehensive introduction," *Natural computing,* vol. 1, no. 1, pp. 3-52, 2002.

[63] J. Liao and R. Bai, "Study on the Performance Support Vector Machine by Parameter Optimized," in *International Conference on Advanced Software Engineering and Its Applications*, 2008, pp. 79-92.

[64] X. Yi, P. Wu, D. Dai, L. Liu, and X. He, "Intrusion Detection Using BP Optimized by PSO," *International Journal of Advancements in Computing Technology,* vol. 4, no. 2, 2012.

[65] H. Saxena and V. Richariya, "Intrusion Detection System using K-means, PSO with SVM Classifier: A Survey," *International Journal of Emerging Technology and Advanced Engineering,* vol. 4, no. 2, pp. 653-657, 2014.

[66] T. N. Report. (2013), "The Nelson Report". available at: http://www.nilsonreport.com/publication_newsletter_archive_issue.php?issue=1023 (accessed 31 July 2014).

[67] T. U. C. Association. "The UK Cards Association". available at: http://www.theukcardsassociation.org.uk/news/EOYFFfor2013.asp (accessed 31 July 2014).

[68] B. C. Johnson. (2003), "Intrusions and their Detection: Addressing Common Hacker Exploits". available at: http://systemexperts.com/media/pdf/hackerid.pdf (accessed 18-September-2014).

[69] J. S. White, J. N. Matthews, and J. L. Stacy, "A method for the automated detection phishing websites through both site characteristics and image analysis," in *SPIE Defense, Security, and Sensing*, 2012, pp. 1-11.

[70] A. Bergholz, J. De Beer, S. Glahn, M.-F. Moens, G. Paaß, and S. Strobel, "New filtering approaches for phishing email," *Journal of computer security*, vol. 18, no. 1, pp. 7-35, 2010.

[71] B. Adida, S. Hohenberger, and R. L. Rivest, "Lightweight encryption for email," in *Proceedings of the USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI 2005)*, 2005, pp. 93-99.

[72] R. Dhamija, J. D. Tygar, and M. Hearst, "Why phishing works," in *Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI)*, Montr´eal, Qu´ebec, Canada, 2006, pp. 581-590.

[73] R. Dhamija and J. D. Tygar, "The battle against phishing: Dynamic security skins," in *Proceedings of the 2005 symposium on Usable privacy and security*, 2005, pp. 77-88.

[74] W. L. Buntine, "A theory of learning classification rules," Doctoral dissertation, School of Computing Science, University of Technology, Sydney, Australia., 1992.

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

[76] S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, and C. Zhang, "An empirical analysis of phishing blacklists," in *Proceedings of Sixth Conference on Email and Anti-Spam (CEAS)*, Mountain View, California USA, 2009, pp. 1-10.

[77] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: a content-based approach to detecting phishing web sites," in *Proceedings of the 16th international conference on World Wide Web*, Banff, Alberta, Canada, 2007, pp. 639-648.

[78] M. Radha Damodaram and M. Valarmathi, "Phishing Website Detection and Optimization Using Particle Swarm Optimization Technique," *International Journal of Computer Science and Security (IJCSS)*, vol. 5, no. 5, pp. 477-490, 2011.

[79] X. Fang, N. Koceja, J. Zhan, G. Dozier, and D. Dipankar, "An artificial immune system for phishing detection," in *2012 IEEE Congress on Evolutionary Computation (CEC)*, 2012, pp. 1-7.

[80] P. A. Barraclough, G. Sexton, M. A. Hossain, and N. Aslam, "Intelligent phishing detection parameter framework for E-banking transactions based on Neuro-fuzzy," in *Science and Information Conference (SAI), 2014*, London, 2014, pp. 545-555.

[81] H. Huang, L. Qian, and Y. Wang, "A SVM-based technique to detect phishing URLs," *Information Technology Journal*, vol. 11, no. 7, pp. 921-925, 2012.

[82] J. Yearwood, M. Mammadov, and A. Banerjee, "Profiling phishing emails based on hyperlink information," in *2010 International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2010, pp. 120-127.

[83] M. Chandrasekaran, K. Narayanan, and S. Upadhyaya, "Phishing email detection based on structural properties," in *Proceedings of the NYS Cyber Security Conference*, 2006, pp. 1-7.

[84] F. Sanchez and D. Zhenhai, "A Sender-Centric Approach to Detecting Phishing Emails," in *2012 International Conference on Cyber Security (CyberSecurity)*, 2012, pp. 32-39.

[85] A. Martin, N. Anutthamaa, M. Sathyavathy, M. M. S. Francois, and P. Venkatesan, "A Framework for Predicting Phishing Websites Using Neural Networks," *International Journal of Computer Science Issues (IJCSI)*, vol. 8, no. 2, pp. 330-336, 2011.

[86] A. Aggarwal, A. Rajadesingan, and P. Kumaraguru, "PhishAri: Automatic realtime phishing detection on twitter," in *eCrime Researchers Summit (eCrime), 2012*, Las Croabas, 2012, pp. 1-12.

[87] H. Zhang, G. Liu, T. W. Chow, and W. Liu, "Textual and visual content-based anti-phishing: a Bayesian approach," *IEEE Transactions on Neural Networks* vol. 22, no. 10, pp. 1532-1546, 2011.

[88] M. Khonji, Y. Iraqi, and A. Jones, "Enhancing Phishing E-Mail Classifiers: A Lexical URL Analysis Approach," *International Journal for Information Security Research (IJISR)*, vol. 3, no. 1, pp. 236-245, 2013.

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

[90] R. Amin, J. J. C. Ryan, and J. R. van Dorp, "Detecting Targeted Malicious Email," *Security & Privacy, IEEE,* vol. 10, no. 3, pp. 64-71, 2012.

[91] D. Debarr, V. Ramanathan, and H. Wechsler, "Phishing detection using traffic behavior, spectral clustering, and random forests," in *2013 IEEE International Conference on Intelligence and Security Informatics (ISI)*, Seattle, WA, 2013, pp. 67-72.

[92] C. Whittaker, B. Ryner, and M. Nazif, "Large-Scale Automatic Classification of Phishing Pages," in *Proceedings of the 17th Network and Distributed System Security Symposium (NDSS)*, 2010, pp. 1-14.

[93] M. Aburrous, M. A. Hossain, K. Dahal, and F. Thabatah, "Modelling Intelligent Phishing Detection System for E-banking Using Fuzzy Data Mining," in *2009 International Conference on CyberWorlds*, Bradford, UK, 2009, pp. 265-272.

[94] F. Toolan and J. Carthy, "Feature selection for Spam and Phishing detection," in *eCrime Researchers Summit (eCrime)*, Dallas, USA, 2010, pp. 1-12.

[95] M. Liping, R. Torney, P. Watters, and S. Brown, "Automatically Generating Classifier for Phishing Email Prediction," in *2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN)*, 2009, pp. 779-783.

[96] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in *Proceedings of the 2007 ACM workshop on Recurring malcode*, Alexandria, Virginia, USA, 2007, pp. 1-8.

[97] R. S. Rao and S. T. Ali, "A Computer Vision Technique to Detect Phishing Attacks," in *2015 Fifth International Conference on Communication Systems and Network Technologies (CSNT)*, Gwalior, India, 2015, pp. 596-601.

[98] P. Likarish, J. Eunjin, D. Dunbar, T. E. Hansen, and J. P. Hourcade, "B-APT: Bayesian Anti-Phishing Toolbar," in *2008. ICC '08. IEEE International Conference on Communications*, Beijing, 2008, pp. 1745-1749.

[99] W. D. Yu, S. Nargundkar, and N. Tiruthani, "Phishcatch-a phishing detection tool," in *33rd Annual IEEE International on Computer Software and Applications Conference*, Seattle, WA, USA, 2009, pp. 451-456.

[100] S. Sathish and A. Thirunavukarasu, "Phishing Webpage Detection for Secure Online Transactions," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 15, no. 3, pp. 86-90, 2015.

[101] R. S. Rao and S. T. Ali, "PhishShield: A Desktop Application to Detect Phishing Webpages through Heuristic Approach," *Procedia Computer Science,* vol. 54, pp. 147-156, 2015.

[102] M. M. Al-Daeef, N. Basir, and M. M. Saudi, "A Method to Measure the Efficiency of Phishing Emails Detection Features," in *2014 International Conference on Information Science and Applications (ICISA),* 2014, pp. 1-5.

[103] I. R. A. Hamid and J. Abawajy, "Phishing Email Feature Selection Approach," in *2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Changsha, 2011, pp. 916-921.

[104] D. L. Cook, V. K. Gurbani, and M. Daniluk, "Phishwish: a simple and stateless phishing filter," *Security and Communication Networks,* vol. 2, no. 1, pp. 29-43, 2009.

[105] N. Sanglerdsinlapachai and A. Rungsawang, "Web phishing detection using classifier ensemble," in *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services*, Paris, France, 2010, pp. 210-215.

[106] F. Toolan and J. Carthy, "Phishing detection using classifier ensembles," in *eCrime Researchers Summit, 2009. eCRIME'09.*, acoma WA USA, 2009, pp. 1-9.

[107] A. Saberi, M. Vahidi, and B. M. Bidgoli, "Learn to Detect Phishing Scams Using Learning and Ensemble ?Methods," in *2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology Workshops*, 2007, pp. 311-314.

[108] S. Marchal, J. Francois, R. State, and T. Engel, "PhishStorm: Detecting Phishing With Streaming Analytics," *IEEE Transactions on Network and Service Management*, vol. 11, no. 4, pp. 458-471, 2014.

[109] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM,* vol. 13, no. 7, pp. 422-426, 1970.

[110] P. Likarish, E. Jung, D. Dunbar, T. E. Hansen, and J. P. Hourcade, "B-apt: Bayesian anti-phishing toolbar," in *IEEE International Conference on Communications*, Beijing, China, 2008, pp. 1745-1749.

[111] M. R. Aburrous, A. Hossain, K. Dahal, and F. Thabatah, "Modelling intelligent phishing detection system for e-banking using fuzzy data mining," in *International Conference on CyberWorlds, 2009. CW'09.* , 2009, pp. 265-272.

[112] 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.

[113] P. Deshmukh, M. Shelar, and N. Kulkarni, "Detecting of targeted malicious email," in *2014 IEEE Global Conference on Wireless Computing and Networking (GCWCN)*, 2014, pp. 199-202.

[114] A. Saberi, M. Vahidi, and B. M. Bidgoli, "Learn to Detect Phishing Scams Using Learning and Ensemble Methods," in *2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, 2007, pp. 311-314.

[115] M. Chandrasekaran, K. Narayanan, and S. Upadhyaya, "Phishing email detection based on structural properties," *NYS Cyber Security Conference*, pp. 1-7, 2006.

[116] O. Kufandirimbwa and R. Gotora, "Spam Detection Using Artificial Neural Networks (Perceptron Learning Rule)," *Online Journal of Physical and Environmental Science Research*, vol. 1, no. 2, pp. 22-29, 2012.

[117] K. R. D and V. P, "A Hybrid ACO Based Feature Selection Method for Email Spam Classification," *WSEAS Transactions on Computers*, vol. 14, pp. 171 - 177, 2015.

[118] B. Yu and Z.-b. Xu, "A comparative study for content-based dynamic spam classification using four machine learning algorithms," *Knowledge-Based Systems*, vol. 21, no. 4, pp. 355-362, 2008.

[119] S. Radicati and Q. Hoang. (2012), "Email statistics report, 2012-2016". available at: http://www.radicati.com/wp/wp-content/uploads/2012/04/Email-Statistics-Report-2012-2016-Executive-Summary.pdf (accessed 22-May-2017).

[120] S. P. Teli and S. Biradar, "Effective Email Classification for Spam and Non-Spam," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 6, pp. 273-278, 2014.

[121] Y. Zhang, S. Wang, P. Phillips, and G. Ji, "Binary PSO with mutation operator for feature selection using decision tree applied to spam detection," *Knowledge-Based Systems*, vol. 64, no. 2014, pp. 22-31, 2014.

[122] S. M. Kalaibar and S. N. Razavi, "Spam filtering by using Genetic based Feature Selection," *International Journal of Computer Applications Technology and Research*, vol. 3, no. 12, pp. 839 - 843, 2014.

[123] D. Evett. (2006), "Spam Statistics 2006". available at: http://www.artefaktum.hu/oktatashoz/internet06osz/05_reklam/spam-statistics.html (accessed 11-July-2017).

[124] G. Bhagyashri, K. H. Pratap, and D.Y.Patil, "Auto E-Mails Classification Using Bayesian Filter," *International Journal of Advanced Technology & Engineering Research (IJATER),* vol. 3, no. 4, pp. 19-24, 2013.

[125] MessageLabs. (2006), "MessageLabs Intelligence: March 2006 ". available at: http://londonactionplan.org/wp-content/uploads/2012/12/mli-report-mar-2006-FINAL.pdf (accessed 02-February-2017).

[126] E. S. M. El-Alfy, "Discovering classification rules for email spam filtering with an ant colony optimization algorithm," in *2009 IEEE Congress on Evolutionary Computation*, Trondheim, Norway, 2009, pp. 1778-1783.

[127] W.-C. Hsu and T.-Y. Yu, "Support vector machines parameter selection based on combined taguchi method and staelin method for e-mail spam filtering," *International Journal of Engineering and Technology Innovation,* vol. 2, no. 2, pp. 113-125, 2012.

[128] B. Issac, W. J. Jap, and J. H. Sutanto, "Improved Bayesian anti-spam filter implementation and analysis on independent spam corpuses," in *2009. ICCET'09. International Conference on Computer Engineering and Technology*, 2009, pp. 326-330.

[129] J. Dudley, "Improving the Performance of Heuristic Spam Detection using a Multi-Objective Genetic Algorithm," Honours Thesis, Computer Science & Software Engineering, The University of Western Australia, 2007.

[130] C.-Y. Tseng and C. Ming-Syan, "Incremental SVM Model for Spam Detection on Dynamic Email Social Networks," *Computational Science and Engineering, 2009. CSE '09.* , vol. 4, pp. 128-135, 2009.

[131] Y. LIU, G. WANG, H. CHEN, Z. ZHAO, X. ZHU, and Z. LIU, "An adaptive fuzzy ant colony optimization for feature selection," *Journal of Computational Information Systems,* vol. 7, no. 4, pp. 1206-1213, 2011.

[132] J. N. Shrivastava and M. H. Bindu, "E-mail spam filtering using adaptive genetic algorithm," *International Journal of Intelligent Systems and Applications (IJISA),* vol. 6, no. 2, p. 54, 2014.

[133] A. Behjat, A. Mustapha, H. Nezamabadi-pour, M. Sulaiman, and N. Mustapha, "GA-based feature subset selection in a spam/non-spam detection system," in *2012 International Conference on Computer and Communication Engineering (ICCCE)*, Kuala Lumpur, Malaysia, 2012, pp. 675-679.

[134] J. N. Shrivastava and M. H. Bindu, "E-mail classification using genetic algorithm with heuristic fitness function," *International Journal of Computer Trends and Technology (IJCTT),* vol. 4, no. 8, pp. 2956 - 2961, 2013.

[135] J. Dudley, L. Barone, and L. While, "Multi-objective spam filtering using an evolutionary algorithm," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, Hong Kong, China, 2008, pp. 123-130.

[136] P. Cortez, R. F. M. Vaz, M. Rocha, M. Rio, and P. Sousa, "Evolutionary symbiotic feature selection for email spam detection," *9th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2012),* vol. 1, no. 9, pp. 159-164, 2012.

[137] A. Abi-Haidar and L. M. Rocha, "Adaptive Spam Detection Inspired by the Immune System," in *Eleventh International Conference on the Simulation and Synthesis of Living Systems, S. Bullock, J. Noble, R. A. Watson, and M.A Bedau (Eds). MIT Press*, 2008, pp. 1-8.

[138] M. Prilepok, T. Jezowicz, J. Platos, and V. Snasel, "Spam detection using compression and PSO," in *2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN)*, 2012, pp. 263-270.

[139] H. Wu, H. z. Li, G. Wang, H. l. Chen, and X. k. Li, "A Novel Spam Filtering Framework Based on Fuzzy Adaptive Particle Swarm Optimization," in *2011 Fourth International Conference on Intelligent Computation Technology and Automation*, 2011, pp. 38-41.

[140] C.-Y. Tseng and M.-S. Chen, "Incremental SVM model for spam detection on dynamic email social networks," in *2009 International Conference on Computational Science and Engineering*, 2009, pp. 128-135.

[141] A. Nosseir, K. Nagati, and I. Taj-Eddin, "Intelligent word-based spam filter detection using multi-neural networks," *International Journal of Computer Science Issues,* vol. 10, no. 2, pp. 17-21, 2013.

[142] C.-H. Wu and C.-H. Tsai, "Robust classification for spam filtering by back-propagation neural networks using behavior-based features," *Applied Intelligence,* vol. 31, no. 2, pp. 107-121, 2009.

[143] D. Vira, P. Raja, and S. Gada, "An Approach to Email Classification Using Bayesian Theorem," *Global Journal of Computer Science and Technology,* vol. 12, no. 13, 2012.

[144] T. A. Almeida, A. Yamakami, and J. Almeida, "Evaluation of Approaches for Dimensionality Reduction Applied with Naive Bayes Anti-Spam Filters," in *2009 International Conference on Machine Learning and Applications*, 2009, pp. 517-522.

[145] A. M. Goweder, T. Rashed, S. Ali, and H. A. Alhammi, "An Anti-spam system using artificial neural networks and genetic algorithms," in *Proceedings of the 2008 International Arab Conference on Information Technology*, 2008, pp. 1-8.

[146] K. M and R. Kumar, "Spam Mail Classification Using Combined Approach of Bayesian and Neural Network," in *2010 International Conference on Computational Intelligence and Communication Networks*, 2010, pp. 145-149.

[147] H. Yin, F. Cheng, and D. Zhang, "Using LDA and ant colony algorithm for spam mail filtering," in *Proceedings of the 2009 Second International Symposium on Information Science and Engineering*, Washington, DC, USA, 2009, pp. 368-371.

[148] Y. Yang, "A Novel Framework Based on Rough Set, Ant Colony Optimization and Genetic Algorithm for Spam Filtering," *International Journal of Advancements in Computing Technology*, vol. 4, no. 14, pp. 516-525, 2012.

[149] R. A. Zitar and A. Hamdan, "Genetic optimized artificial immune system in spam detection: a review and a model," *Artificial Intelligence Review*, vol. 40, no. 3, pp. 305-377, 2013.

[150] K. R. Dhanaraj and V. Palaniswami, "Firefly and BAYES Classifier for Email Spam Classification in a Distributed Environment," *Australian Journal of Basic and Applied Sciences*, vol. 8, no. 17, pp. 118-130, 2014.

[151] X. Wang, J. Yang, X. Teng, W. Xia, and R. Jensen, "Feature selection based on rough sets and particle swarm optimization," *Pattern Recognition Letters*, vol. 28, no. 4, pp. 459-471, 2007.

[152] S. Salehi and A. Selamat, "Hybrid simple artificial immune system (SAIS) and particle swarm optimization (PSO) for spam detection," in *2011 Malaysian Conference in Software Engineering*, 2011, pp. 124-129.

[153] X. l. Chen, P. y. Liu, Z. f. Zhu, and Y. Qiu, "A method of spam filtering based on weighted support vector machines," in *2009 IEEE International Symposium on IT in Medicine & Education*, 2009, pp. 947-950.

[154] B. Issac, W. J. Jap, and J. H. Sutanto, "Improved Bayesian Anti-Spam Filter Implementation and Analysis on Independent Spam Corpuses," in *2009 International Conference on Computer Engineering and Technology*, 2009, pp. 326-330.

[155] A. S. Foundation. "Apache Software Foundation. Spamassassin homepage". available at: http://spamassassin.apache.org/ (accessed 05-August-2014).

[156] G. V. Cormack and T. R. Lynam, "Spam Corpus Creation for TREC," in *Conference on Email and Anti-Spam (CEAS)*, 2005.

[157] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, *et al.*, "Map-reduce for machine learning on multicore," in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, Canada, 2006, pp. 281-288.

[158] S. Salehi and A. Selamat, "Hybrid simple artificial immune system (SAIS) and particle swarm optimization (PSO) for spam detection," in *2011 5th Malaysian Conference in Software Engineering (MySEC)*, 2011, pp. 124-129.

[159] W. W. Cohen, "Learning rules that classify e-mail," in *AAAI spring symposium on machine learning in information access*, 1996, pp. 18-25.

[160] X. Carreras and L. Marquez, "Boosting trees for anti-spam email filtering," in *4th International Conference on Recent Advances in Natural Language Processing*, Tzigov Chark, Bulgaria, 2001, pp. 58–64.

[161] M. Zareapoor, S. KR, and M. A. Alam, "Analysis of Credit Card Fraud Detection Techniques: based on Certain Design Criteria," *International Journal of Computer Applications,* vol. 52, no. 3, pp. 35-42, 2012.

[162] S. Maes, K. Tuyls, B. Vanschoenwinkel, and B. Manderick, "Credit card fraud detection using Bayesian and neural networks," in *Proceedings of the 1st international naiso congress on neuro fuzzy technologies*, 2002, pp. 261-270.

[163] J. K.-F. Pun, "Improving Credit Card Fraud Detection using a Meta-Learning Strategy," Master of Applied Science, Graduate Department of Chemical Engineering and Applied Chemistry University of Toronto, 2011.

[164] B. Insider. (2015), "Payments companies are trying to fix the massive credit-card fraud problem with these 5 new security protocols". available at: http://www.businessinsider.com/how-payment-companies-are-trying-to-close-the-massive-hole-in-credit-card-security-2015-3 (accessed 01-December-2015).

[165] N. Wong, P. Ray, G. Stephens, and L. Lewis, "Artificial immune systems for the detection of credit card fraud: an architecture, prototype and preliminary results," *Information Systems Journal*, vol. 22, no. 1, pp. 53-76, 2012.

[166] VISA. (2005a), "Fact sheet 12: security and fraud prevention". available at: http://www.visa-asia.com/ap/au/mediacenter/factsheets/includes/uploads/FS12_security_amd_prevention.pdf (accessed 02-December-2015).

[167] FFA. (2015), "Stop and spot: cardnot-present fraud". available at: http://www.financialfraudaction.org.uk/ (accessed 02-December-2015).

[168] Y. Sahin and E. Duman, "Detecting credit card fraud by decision trees and support vector machines," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Hong Kong, 2011, pp. 1-6.

[169] S. Ehramikar, "The Enhancemeat of Credit Card Fraud Detectioa Systems using Machine Learning Methodology," Masters, Department of Chernical Engineering and Applied Chemistry, University of Toronto, 2000.

[170] E. Duman and M. H. Ozcelik, "Detecting credit card fraud by genetic algorithm and scatter search," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13057-13063, 2011.

[171] J. T. Quah and M. Sriganesh, "Real-time credit card fraud detection using computational intelligence," *Expert Systems with Applications*, vol. 35, no. 4, pp. 1721-1732, 2008.

[172] M. Z. Khan, J. D. Pathan, and A. H. E. Ahmed, "Credit Card Fraud Detection System Using Hidden Markov Model and K-Clustering," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 3, no. 2, pp. 5458-5461, 2014.

[173] A. P. Khan, V. S. Mahajan, S. H. Shaikh, and A. B. Koli, "Credit Card Fraud Detection System through Observation Probability Using Hidden Markov Model," *International Journal of Thesis Projects and Dissertations (IJTPD)*, vol. 1, no. 1, pp. 7-16, 2013.

[174] N. Soltani, M. K. Akbari, and M. S. Javan, "A new user-based model for credit card fraud detection based on artificial immune system," in *The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012)*, Shiraz, Fars, Iran, 2012, pp. 29-33.

[175] N. Soltani Halvaiee and M. K. Akbari, "A novel model for credit card fraud detection using Artificial Immune Systems," *Applied Soft Computing*, vol. 24, pp. 40-49, 2014.

[176] R. D. Patel and D. K. Singh, "Credit Card Fraud Detection & Prevention of Fraud Using Genetic Algorithm," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 6, pp. 2231-2307, 2013.

[177] K. RamaKalyani and D. UmaDevi, "Fraud Detection of Credit Card Payment System by Genetic Algorithm," *International Journal of Scientific & Engineering Research*, vol. 3, no. 7, pp. 1-6, 2012.

[178] H. Modi, S. Lakhani, N. Patel, and V. Patel, "Fraud Detection in Credit Card System Using Web Mining," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, no. 2, pp. 175-179, 2013.

[179] V. Van Vlasselaer, C. Bravo, O. Caelen, T. Eliassi-Rad, L. Akoglu, M. Snoeck, *et al.*, "APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions," *Decision Support Systems*, vol. 75, pp. 38-48, 2015.

[180] S. Mhamane and L. Lobo, "Fraud Detection in Online Banking Using HMM," in *International Conference on Information and Network Technology (ICINT 2012)*, Singapore, 2012, pp. 200-204.

[181] V. Bhusari and S. Patil, "Study of Hidden Markov Model in Credit Card Fraudulent Detection," *International Journal of Computer Applications (0975–8887)*, vol. 20, no. 5, pp. 33-36, 2011.

[182] S. S. Mhamane and L. J. Lobo, "Use of Hidden Markov Model as Internet Banking Fraud Detection," *International Journal of Computer Applications*, vol. 45, no. 21, pp. 5-10, 2012.

[183] S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan, "Credit card fraud detection using meta-learning: Issues and initial results," *AAAI-97 Workshop on Fraud Detection and Risk Management*, 1997.

[184] S. K. Sen and S. Dash, "Meta Learning Algorithms for Credit Card Fraud Detection," *Meta*, vol. 6, no. 6, pp. 16-20, 2013.

[185] Q. Lu and C. Ju, "Research on credit card fraud detection model based on class weighted support vector machine," *Journal of Convergence Information Technology*, vol. 6, no. 1, pp. 295-298, 2011.

[186] K. Seeja and M. Zareapoor, "FraudMiner: A Novel Credit Card Fraud Detection Model Based on Frequent Itemset Mining," *The Scientific World Journal,* vol. 2014, p. 10 Pages, 2014.

[187] G. Potamitis, "Design and Implementation of a Fraud Detection Expert System using Ontology-Based Techniques," Master of Science School of Computer Science, University of Manchester, 2013.

[188] M. Carminati, R. Caron, F. Maggi, I. Epifani, and S. Zanero, "BankSealer: A decision support system for online banking fraud analysis and investigation," *Computers & Security,* vol. 53, pp. 175-186, 2015.

[189] N. Mahmoudi and E. Duman, "Detecting credit card fraud by Modified Fisher Discriminant Analysis," *Expert Systems with Applications,* vol. 42, no. 5, pp. 2510-2516, 2015.

[190] A. U. S. Khan, N. Akhtar, and M. N. Qureshi, "Real-Time Credit-Card Fraud Detection using Artificial Neural Network Tuned by Simulated Annealing Algorithm," in *Conf. on Recent Trends in Information, Telecommunication and Computing, ITC*, 2014, pp. 113-121.

[191] K. Ganesh and P. V. Sena, "Novel Artificial Neural Networks and Logistic Approach for Detecting Credit Card Deceit," *International Journal of Computer Science and Network Security,* vol. 13, no. 9, pp. 58-65, 2012.

[192] M. Zareapoor and P. Shamsolmoali, "Application of credit card fraud detection: Based on bagging ensemble classifier," *Procedia Computer Science,* vol. 48, pp. 679-685, 2015.

[193] S. Jha, M. Guillen, and J. C. Westland, "Employing transaction aggregation strategy to detect credit card fraud," *Expert systems with applications,* vol. 39, no. 16, pp. 12650-12657, 2012.

[194] A. U. S. Khan, N. Akhtar, and M. N. Qureshi, "Real-time credit-card fraud detection using artificial neural network tuned by simulated annealing algorithm," in *Proceedings of International Conference on Recent Trends in Information, Telecommunication and Computing*, Chandigarh, India, 2014, pp. 113-121.

[195] R. Damodaram and M. Valarmathi, "Phishing website detection and optimization using modified bat algorithm," *Int. J. Engineering Research and Applications,* vol. 2, no. 1, pp. 870-876, 2012.

[196] M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: a literature survey," *Communications Surveys & Tutorials, IEEE,* vol. 15, no. 4, pp. 2091-2121, 2013.

[197] V. Vapnik, *The nature of statistical learning theory*: Springer-Verlag New York, 1995.

[198] S. Ertekin, "Learning in extreme conditions: Online and active learning with massive, imbalanced and noisy data," Doctor of Philosophy, Computer Science and Engineering, The Pennsylvania State University, 2009.

[199] H.-T. KUHN, "Nonlinear programming," in *J. Neyman ed., Proceedings of the Second Berkeley Symposium on Mathematical Sta- tistics and Probability (Berkeley: University of California Press, 1951)*, 1951, pp. 481-492.

[200] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine learning,* vol. 20, no. 3, pp. 273-297, September, 1995.

[201] W. Karush, "Minima of functions of several variables with inequalities as side conditions," Masters Degree, Dept. of Mathematics, University of Chicago, Chicago, Illinois, 1939.

[202] S. Alexander, F. Constantin, P. Douglas, and G. Isabelle, *A Gentle Introduction to Support Vector Machines in Biomedicine* vol. 1: Theory and Methods. Singapore: World Scientific Press, 2011.

[203] J. Nocedal and S. Wright, *Numerical Optimization*. New York: Springer Science & Business Media, 2006.

[204] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," *Advances in kernel methods: support vector learning, MIT Press, Cambridge, MA,* pp. 185-208, 1999.

[205] O. Chapelle, "Training a support vector machine in the primal," *Neural computation,* vol. 19, no. 5, pp. 1155-1178, 2007.

[206] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine learning,* vol. 38, no. 3, pp. 257-286, 2000.

[207] C. Stanfill and D. Waltz, "Toward memory-based reasoning," *Communications of the ACM,* vol. 29, no. 12, pp. 1213-1228, 1986.

[208] D. Wettschereck, "A hybrid nearest-neighbor and nearest-hyperrectangle algorithm," in *European Conference on Machine Learning,* 1994, pp. 323-335.

[209] I. Watson and F. Marir, "Case-based reasoning: A review," *Knowledge Engineering Review,* vol. 9, no. 4, pp. 327-354, 1994.

[210] D. Wettschereck and T. G. Dietterich, "An experimental comparison of nearest-neighbor and nearesthyperrectangle

algorithms," *Machine Learning*, vol. 19, no. 1, pp. 5-28, 1995.

[211] MathWorks. (2016), "Edge Detection". available at: http://www.mathworks.com/discovery/edge-detection.html (accessed 02-June-2016).

[212] D. Ray. (2013), "Edge Detection in Digital Image Processing". available at: https://www.math.washington.edu/~morrow/336_13/papers/debosmit.pdf (accessed 30-June -2016).

[213] P. H. King, "Digital Image Processing and Analysis: Human and Computer Applications with CVIPtools, 2nd Edition (Umbaugh, S.; 2011) [Book Reviews]," *IEEE Pulse*, vol. 3, no. 4, pp. 84-85, 2012.

[214] M. Dorigo, "Optimization, learning and natural algorithms," PhD Thesis PhD Thesis, Politecnico di Milano, Italy, 1992.

[215] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theoretical computer science*, vol. 344, no. 2, pp. 243-278, 2005.

[216] S. Katiyar and A. Q. Ansari, "Ant colony optimization: A tutorial review," *MR International Journal of Engineering & Technology*, vol. 7, no. 2, pp. 35-41, 2015.

[217] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28-39, 2006.

[218] M. D. Toksari, "A hybrid algorithm of Ant Colony Optimization (ACO) and Iterated Local Search (ILS) for estimating electricity domestic consumption: Case of Turkey," *International Journal of Electrical Power & Energy Systems*, vol. 78, pp. 776-782, 2016.

[219] M. Batouche and S. Meshoul, "Nature Inspired Intelligent Techniques for Problem Solving, Technical Report, King Saud University, Riyadh, Kingdom of Saudi Arabia," 2010.

[220] K. L. Hoffman, M. Padberg, and G. Rinaldi, "Traveling Salesman Problem," in Encyclopedia of Operations Research and Management Science, S. I. Gass and M. C. Fu, Eds., ed Boston, MA: Springer US, 2013, pp. 1573-1578.

[221] P. Rabanal, I. Rodríguez, and F. Rubio, "Using river formation dynamics to design heuristic algorithms," in *International Conference on Unconventional Computation*, 2007, pp. 163-177.

[222] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," *Statistical science*, vol. 8, no. 1, pp. 10-15, 1993.

[223] X.-S. Yang, "Flower pollination algorithm for global optimization," in *International Conference on Unconventional Computing and Natural Computation*, 2012, pp. 240-249.

[224] J. James and V. O. Li, "A social spider algorithm for global optimization," *Applied Soft Computing*, vol. 30, pp. 614-627, 2015.

[225] A. R. Jordehi and J. Jasni, "Particle swarm optimisation for discrete optimisation problems: a review," *Artificial Intelligence Review,* vol. 43, no. 2, pp. 243-258, 2015.

[226] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *2009. NaBIC 2009. World Congress on Nature & Biologically Inspired Computing*, 2009, pp. 210-214.

[227] X.-S. Yang, "A New Metaheuristic Bat-Inspired Algorithm," in 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, 2010, pp. 65-74.

[228] 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).

[229] L. Chittka, J. D. Thomson, and N. M. Waser, "Flower constancy, insect psychology, and plant evolution," *Naturwissenschaften,* vol. 86, no. 8, pp. 361-377, 1999.

[230] R. F. Foelix, *Biology of spiders: Oxford University Press*. Stuttgart: Georg Thieme Verlag., 1979.

[231] F. F. Campón, "Group foraging in the colonial spider Parawixia bistriata (Araneidae): effect of resource levels and prey size," *Animal Behaviour,* vol. 74, no. 5, pp. 1551-1562, 2007.

[232] C. F. Schaber, S. N. Gorb, and F. G. Barth, "Force transformation in spider strain sensors: white light interferometry," *Journal of The Royal Society Interface,* vol. 9, no. 71, pp. 1254-1264, 2012.

[233] J. S. House, K. R. Landis, and D. Umberson, "Social relationships and health," *Science*, vol. 241, no. 4865, pp. 540-545, 1988.

[234] C. W. Clark and M. Mangel, "Foraging and flocking strategies: information in an uncertain environment," *American Naturalist*, pp. 626-641, 1984.

[235] C. J. Barnard and R. M. Sibly, "Producers and scroungers: a general model and its application to captive flocks of house sparrows," *Animal behaviour,* vol. 29, no. 2, pp. 543-550, 1981.

[236] G. Beauchamp, "Group foraging revisited: information sharing or producer-scrounger game?," *The American Naturalist,* vol. 148, no. 4, pp. 738-743, 1996.

[237] James-Yu. (2015), "Social Spider Algorithm". available at: https://github.com/James-Yu/SocialSpiderAlgorithm/blob/master/MATLAB/SSA.m (accessed 20-September-2016).

[238] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*. University of Cambridge, United Kingdom: Luniver Press, 2008.

[239] I. Fister, I. JrFister, X.-S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation,* vol. 13, pp. 34-46, 2013.

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

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

[242] R. Shams and R. E. Mercer, "Classifying spam emails using text and readability features," in *2013 IEEE 13th international conference on data mining*, 2013, pp. 657-666.

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

[244] 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.

[245] 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, pp. 1099–1107, 2012.

[246] N. Zhang and Y. Yuan, "CS229 lecture notes, Phishing Detection Using Neural Network," Department of Statistics, Stanford University, 2012, available.at: http://cs229.stanford.edu/proj2012/ZhangYuan-PhishingDetectionUsingNeuralNetwork.pdf. (accessed 10-July-2017).

[247] R. Basnet, S. Mukkamala, and A. H. Sung, "Detection of Phishing Attacks: A Machine Learning Approach," in Soft Computing Applications in Industry, B. Prasad, Ed., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 373-383.

[248] M. Khonji, Y. Iraqi, and A. Jones, "Phishing Detection: A Literature Survey," *IEEE Communications on Surveys & Tutorials*, vol. 15, no. 4, pp. 2091-2121, 2013.

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

[250] K. Seeja and M. Zareapoor, "FraudMiner: a novel credit card fraud detection model based on frequent itemset mining," *The Scientific World Journal*, vol. vol. 2014, Article ID 252797, 10 pages, 2014.

[251] 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, 2010.

[252] C.-H. Chou, B.-H. Kuo, and F. Chang, "The generalized condensed nearest neighbor rule as a data reduction method," in *18th International Conference on Pattern Recognition (ICPR'06)*, 2006, pp. 556-559.

[253] 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, 2005.

[254] D. L. Wilson, "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 2, no. 3, pp. 408-421, 1972.

[255] D. R. Wilson and T. R. Martinez, "Instance Pruning Techniques," in *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997, pp. 403-411.

[256] H. Brighton and C. Mellish, "On the Consistency of Information Filters for Lazy Learning Algorithms," in Principles of Data Mining and Knowledge Discovery: Third European Conference, PKDD'99, Prague, Czech Republic, September 15-18, 1999. Proceedings, J. M. Żytkow and J. Rauch, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 283-288.

[257] N. Panda, E. Y. Chang, and G. Wu, "Concept boundary detection for speeding up SVMs," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 681-688.

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

[259] A. P. Piotrowski, M. J. Napiorkowski, J. J. Napiorkowski, and P. M. Rowinski, "Swarm Intelligence and Evolutionary Algorithms: Performance versus speed," *Information Sciences,* vol. 384, pp. 34-85, 2017.

[260] 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 Proceedings 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, 2017, pp. 603-614.