Applied Soft Computing Journal 91 (2020) 106229

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Spam filtering using a logistic regression model trained by an artificial bee colony algorithm

Bilge Kagan Dedeturk [a,*], Bahriye Akay [b]

[a] Erciyes University, Institute of Natural and Applied Sciences, Melikgazi, 38039 Kayseri, Turkey
[b] Erciyes University, Department of Computer Engineering, Melikgazi, 38039 Kayseri, Turkey

ARTICLE INFO

Article history:
Received 27 June 2019
Received in revised form 6 March 2020
Accepted 8 March 2020
Available online 16 March 2020

Keywords:
Spam filtering
Artificial bee colony
Naive Bayes
Support vector machines
Logistic regression
Turkish emails

ABSTRACT

Email spam is a serious problem that annoys recipients and wastes their time. Machine-learning methods have been prevalent in spam detection systems owing to their efficiency in classifying mail as solicited or unsolicited. However, existing spam detection techniques usually suffer from low detection rates and cannot efficiently handle high-dimensional data. Therefore, we propose a novel spam detection method that combines the artificial bee colony algorithm with a logistic regression classification model. The empirical results on three publicly available datasets (Enron, CSDMC2010, and TurkishEmail) show that the proposed model can handle high-dimensional data thanks to its highly effective local and global search abilities. We compare the proposed model's spam detection performance to those of support vector machine, logistic regression, and naive Bayes classifiers, in addition to the performance of the state-of-the-art methods reported by previous studies. We observe that the proposed method outperforms other spam detection techniques considered in this study in terms of classification accuracy.

1. Introduction

Ongoing increases in the number of e-commerce companies have caused the number of advertising emails to increase. Online shopping customers receive emails from unreliable senders to phish their passwords or bank account information. These unsolicited bulk emails, which are sent indiscriminately, are called spam [1]. According to Radicati [2], more than 293 billion emails will be sent and received daily in 2019, and it is forecast that the number of emails sent and received will exceed 347 billion by the end of 2023. Based on [3], spam messages accounted for 55% of worldwide email traffic in 2017 and 2018. Symantec states that overall spam rates in the years 2015, 2016, and 2017 were 52.7%, 53.4%, and 54.6%, respectively [4]. Kaspersky also reported the proportion of spam emails as 53.49% by the end of third quarter of 2018 [5]. Spam email inconveniences recipients, wastes network bandwidth, file storage space, and computational sources, and may infect files by including malware products in attachments [6].

Many methods have been developed to prevent spam. Bhowmick and Hazarika [6] categorized the methods used for mitigating spam emails into eight groups: heuristic filters, blacklisting, whitelisting, challenge response systems, collaborative spam filtering, honey pots, signature schemes, and machine-learning based methods. A more generic categorization divides the methods into two broad groups: static and dynamic [7]. Static methods include filtering approaches based on predefined whitelists and blacklists. Because spam senders use different email addresses and users of infected computers may send spam unaware of the infection, static listing methods are ineffective. Dynamic methods generally consider email content using text classification approaches established with statistical techniques or machine-learning methods to decide whether an email is spam.

Text classification assigns a class $c_j \in C = \{c_1, c_2, \ldots, c_{|C|}\}$ to a document $d_i \in D = \{d_1, d_2, \ldots, d_{|D|}\}$, where C is a set of categories and D is a set of documents. A document should be expressed by an appropriate representation using preprocessing techniques before it is given to a classification algorithm. The preprocessing techniques represent words in the text as a compact and informative vector using tokenization, pruning, stop word removal, stemming, and feature extraction steps. Then, feature selection can be performed to reduce the dimension of the feature set. A function $\phi : D \times C \to \{T, F\}$ returns true (T) if a document d_i is assigned to a class c_j and returns false (F) otherwise. In spam detection, we are given a set of emails $D = \{d_1, d_2, \ldots, d_m\}$ and two classes $C = \{c_{spam}, c_{legitimate}\}$. The objective is to assign email $d_i \in D$ to one of the two classes correctly.

In this study, we propose a novel spam filtering approach that retains the advantages of logistic regression (LR)—simplicity, fast

* Corresponding author.
E-mail addresses: kagandedeturk@gmail.com (B.K. Dedeturk), bahriye@erciyes.edu.tr (B. Akay).

https://doi.org/10.1016/j.asoc.2020.106229
1568-4946/© 2020 Elsevier B.V. All rights reserved.

classification in real-time applications [8], and efficiency—while avoiding its convergence to poor local minima by training it using the artificial bee colony (ABC) [9] algorithm, which is a nature-inspired swarm intelligence algorithm that simulates the foraging behavior of honey bees. This study develops, to best of our knowledge, the first spam filter that uses the ABC as an LR learning algorithm. The ABC algorithm performs well on multimodal and high-dimensional problems [10,11] and exhibits balanced exploration and exploitation ability, which makes it a good option for spam classification. The proposed algorithm reduces another limitation of LR, associated with its sensitivity to equal feature weights, through integration with the term frequency — inverse document frequency ($tf - idf$) method.

We compare the classification accuracy of the proposed model with those of linear support vector machine (SVM), radial basis SVM, Gaussian naive Bayes (NB), multinomial NB, and LR classifiers. Experimental results show that ABC-LR performs better than the other methods on the publicly available datasets CSDMC2010 and TurkishEmail. The effect of language on classification performance is evaluated. The Enron dataset is also used to compare the classification performance of ABC-LR with that of the methods in previous studies; we demonstrate that the proposed approach outperforms state-of-the-art methods previously tested on the Enron dataset in terms of its classification accuracy and false negative (FN) rate. The experiments are repeated for various values of predetermined control parameters to show the classifiers' sensitivity to the parameter values.

Although there are many successful studies for filtering spam email written in English, there are few such studies [7,12] for those written in Turkish, which is an agglutinative language that has a much more complex morphological structure than English [7]. Turkish datasets require special preprocessing techniques to obtain state-of-the-art results. Therefore, this study is important because it examines how to handle issues of language differences and complexity in spam filtering.

The paper is organized as follows: Section 2 presents a literature review; Section 3 briefly introduces the machine learning algorithms used in this study; Section 4 describes the proposed methodology and its constituent frameworks; Section 5 explains the algorithm's datasets, data preprocessing operations, and feature selection method; Section 6 lays out experimental results and discussions, and Section 7 concludes the paper.

2. Literature review

In this paper, we investigated three commonly used machine-learning methods: NB, SVM, and LR. NB classifiers are probabilistic classifiers that reduce a multivariate problem into a group of univariate problems based on a naive assumption that every pair of features in a feature vector is independent [13]. NB is commonly used in spam filtering because of its simplicity, quick convergence, linear computational complexity, and ease of interpretation [13–15]. Androutsopoulos et al. [16] showed that NB outperforms keyword-based filtering. They stated that automatically trainable anti-spam filters are viable and can have a significant effect on classification performance. However, Rusland et al. [17] concluded that the type of email and the number of instances in a dataset affect the performance of NB. NB classifiers can give good accuracy when a dataset has fewer instances of emails and fewer attributes. Almeida et al. [18] presented a study on term-selection methods in dimensionality reduction for spam filtering domain based on Bayesian theory and showed that the performance of NB is highly sensitive to the selected attributes and the number of term-selection methods. Feng et al. [19] reported that the practicability of NB is limited because it assumes the independence of features in the training set, which is usually not true in real-world applications.

Another popular spam filtering technique is an SVM classifier, which analyzes data and identifies the patterns used for categorization. SVMs explore the relationships between the variables of interest by solving quadratic problems and maximize the margin distance between positive and negative data points that are closest to the decision hyperplane [20]. SVM classifiers achieve high accuracy in text classification and spam filtering [21–23]. Amayri and Bouguila [22] performed a detailed study of SVM classifiers for spam filtering and proposed various string and distance-based kernels. They reported that an active online method using string kernels produces higher precision and recall rates. Yu and Xu [24] applied SVMs to spam classification and highlighted the classification accuracy and fast testing times, despite their complexity. [25] proposed two SVM approaches: the first is incrementally trained, and the second heuristically updates the feature set to improve the false positive (FP) rate of a conventional SVM. Skulley and Wachman [23] proposed an online SVM that relaxes the maximum margin requirement to reduce the computational cost of SVM and produces a nearly equivalent result. They point out that the success of their method depends on the nature of the spam data. A survey prepared by Bhowmick and Hazarika [6] indicated that SVMs achieve high accuracy but suffer from high computational costs and FP rates.

LR minimizes the error associated with the output calculated by a logistic activation function. It has also been applied to email classification and demonstrated good performance in spam filtering [8,26,27]. Goodman and Yih [26] proposed a simple LR model trained by an online gradient descent algorithm. They reported that their model can produce competitive results compared to the best reported generative method (NB). Han et al. [8] presented an improved LR model that reduces the sensitivity of basic LR to equal adjustment of the feature weights during the training period. Chang et al. [27] developed a hybrid LR and NB model, called a partitioned LR model. The model decomposes the original feature space into several groups and then applies LR to each group. These predictions are incorporated using the NB rules to obtain a result. The proposed model outperformed both NB and LR on synthetic and real spam filtering data. Chang et al. pointed out the gap related to tuning the smoothing parameter associated with each individual model.

Ozgur et al. [7] used a stemming operator specifically for Turkish and applied NB and single- and multilayer perceptrons (MLPs) to classify Turkish emails, achieving a success rate of approximately 90%. Tunga and Ciltik [12] proposed methods and heuristics based on the n-gram method and the first n-word heuristic. Their best success rate with the approach obtained was approximately 97% for a Turkish dataset.

Recently, nature-inspired algorithms have become popular due to their highly adaptive features. Idris and Selamat [28] suggested a model that combines a negative selection algorithm (NSA), which is one of the main algorithms in the artificial immune system (AIS), with particle swarm optimization (PSO), and they demonstrated that their proposed model could detect spam emails better than the NSA and PSO models alone. Ramdane and Salim [29] proposed a novel approach that combines NSA with k- means clustering and fruit fly optimization to improve the efficiency of NSA. They empirically showed that their approach outperformed the NSA and NSA-PSO with respect to positive prediction, detection accuracy, and computational complexity. Azam et al. [30] used NSA for anomaly detection and observed that including more datasets improved classification performance. In another study, Naem et al. [31] proposed a method called ALO-Boosting, which consists of two main stages: In the first stage, an ant lion optimizer (ALO) was used to obtain the optimum feature subset; in the second stage, a boosting classifier was used to classify emails as spam or normal. This method achieved considerable success in terms of classification accuracy.

Overall, although these classifiers are easy to implement, generalizable, and fairly efficient, they usually suffer from limitations, such as the curse of dimensionality, high computational costs, misclassification rates, sensitivity to feature weights, slow operation speeds for real applications, and overfitting or getting stuck in local minima. Moreover, the performance of these classifiers depends on some predetermined parameters and the problem type.

Considering the limitations of existing spam detection methods, we propose a novel spam filter approach to classifying an email as spam. The main contribution of this study can be summarized as follows: (1) The proposed model overcomes highly nonlinear and high-dimensional data due to its training algorithm, which performs local and global searches in the solution space and enables the learning of highly complex features from data; (2) it is effective for a wide range of spam datasets: linear and nonlinear, balanced and imbalanced, personalized and non-personalized, and English and Turkish; (3) it deals with language differences and complexity on publicly available datasets; (4) it handles the data imbalance problem, which decreases classification accuracy, of the Enron dataset using a training over sampling method; (5) it reduces the limitation associated with sensitivity to feature weights, through integration with the tf-idf method.

3. Brief explanations of the machine learning algorithms used in the study

Spam filters based on machine-learning algorithms automatically classify a message as spam or legitimate based on its content [32]. In this section, we briefly explain the machine-learning algorithms used herein.

3.1. Naive Bayes (NB) methods

NB methods are probabilistic learning algorithms established on Bayes' theorem, with naive assumptions that every pair of features in a feature vector is independent. According to Bayes' theorem, the probability that the feature vector of an email, $\vec{x} = [x_1, x_2, \dots, x_n]$, belongs to a class, c_j, is computed by using Eq. (1).

$$p(c_j|\vec{x}) = \frac{p(c_j).p(\vec{x}|c_j)}{p(\vec{x})} \tag{1}$$

Because the denominator of Eq. (1) is a constant, NB chooses the class that maximizes $p(c_j).p(\vec{x}|c_j)$. In the case of spam filtering, there are two classes: spam and legitimate. If $p(c_{spam}|\vec{x}) > p(c_{ham}|\vec{x})$, NB classifies feature vector $\vec{x}$ as spam; otherwise, it is classified as ham. Prior probability $p(c_j)$ is the number of training messages in class c_j divided by the total number of training messages. Estimation of probabilities $p(\vec{x}|c_j)$ differs according to the NB method used. These methods are briefly described in the following subsections.

3.1.1. Gaussian NB

Gaussian NB assumes that each feature is normally (Gaussian) distributed and $p(\vec{x}|c_j)$ values are computed by using Eq. (2).

$$p(\vec{x}|c_j) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_{i,j}^2}} e^{-\frac{(x_i-\mu_{i,j})^2}{2\sigma_{i,j}^2}} \tag{2}$$

The parameters' mean $\mu_{i,j}$ and deviation $\sigma_{i,j}$ are estimated by applying maximum likelihood to the training data.

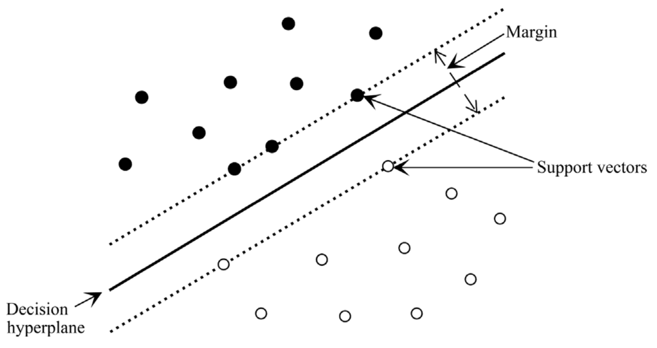

Fig. 1. SVM classifier.

3.1.2. Multinomial NB

In multinomial NB, an email is represented by a feature vector $\vec{x} = [x_1, x_2, \dots, x_n]$, where each x_i is the number of times corresponding term t_i exists in the email. Then, probability $p(\vec{x}|c_j)$ is estimated using Eq. (3).

$$p(\vec{x}|c_j) = \prod_{i=1}^{n} p(t_i|c_j), \tag{3}$$

where $p(t_i|c_j)$ is obtained using Eq. (4).

$$p(t_i|c_j) = \frac{N_{t_i,c_j} + \alpha}{N_{c_j} + \alpha n}, \tag{4}$$

where N_{t_i,c_j} is the number of times term t_i occurs in the training set of class c_j, $N_{c_j} = \sum_{i=1}^{n} N_{t_i,c_j}$ is the total count of all terms in a class c_j, and α is smoothing parameter. When $\alpha = 1$, we refer to the smoothing as Laplacian smoothing, whereas when $\alpha < 1$, it is called Lidstone smoothing.

3.2. Support vector machine (SVM)

An SVM is a popular machine learning algorithm that is widely used in text classification. As seen in Fig. 1, the aim of an SVM is to maximize the margin between the positive and negative data points that are closest to the decision hyperplane. These points are called support vectors.

Given a training set $\{(\vec{x}_1, y_1), \dots, (\vec{x}_m, y_m)\}$, $\vec{x}_i \in R^n$ and $y_i \in \{-1, 1\}$, $1 \le i \le m$, an SVM determines weight vector $\vec{w}$ and bias term b, yielding the best decision hyperplane by solving the optimization problem defined using Eq. (5):

$$\begin{array}{ll} \min & \frac{1}{2}\vec{w}^T\vec{w} \\ \text{subject to} & y_i(\vec{w}^T\vec{x}_i + b) \ge 1 \end{array}. \tag{5}$$

New data points $\vec{x}_j$ can be classified by using Eq. (6):

$$f(\vec{x}_j) = sign(\vec{w}^T\vec{x}_j + b). \tag{6}$$

3.2.1. Linear SVM

In a linear SVM classifier, if a training set is not linearly separable, then the SVM uses a soft margin. In that case, the decision hyperplane separates data points in the training set correctly, though some data points can be located in the wrong class (Fig. 2). To find this type of hyperplane, the SVM introduces a penalty parameter C and a slack variable ξ_i, which corresponds to the cost of misclassified examples. If a sample is assigned to the true class, then the slack variable equals zero. The formulation of the soft

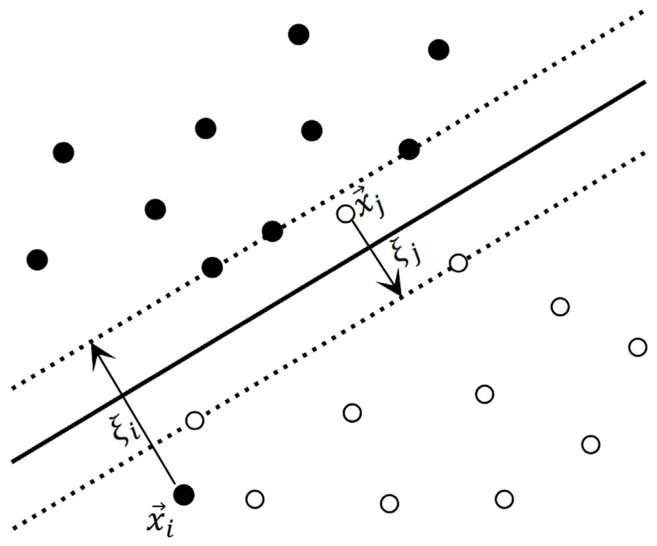

Fig. 2. Linear SVM classifier in a case that is not linearly separable.

margin SVM classifier is given by Eq. (7):

$$\begin{aligned} &\min \quad \frac{1}{2}\vec{w}^T\vec{w} + C\sum_{i=1}^{m}\xi_i \\ &\text{subject to} \quad y_i(\vec{w}^T\vec{x}_i + b) \geq 1 - \xi_i \\ &\qquad\qquad \xi_i \geq 0. \end{aligned} \tag{7}$$

3.2.2. SVM With radial basis function kernel

When the data have nonlinear interactions and the features in feature space have nonlinear dependence, employing linear separators and nonlinear decision boundaries causes inefficient clustering. In this case, SVMs are integrated with kernel functions (Eq. (8)) to adapt to nonlinearities in data.

$$f(x) = sign\left\{\sum_{i=1}^{N}(w_i y_i K(x_i \cdot x_j) + b)\right\} \tag{8}$$

In this study, a radial basis function (RBF) defined by Eq. (9) is used:

$$K_{RBF}(x_i, x_j) = exp(-\gamma\|x_i - x_j\|^2), \gamma > 0. \tag{9}$$

3.3. Logistic regression (LR)

Given a training set $\{(\vec{x}_1, y_1), \ldots, (\vec{x}_m, y_m)\}$, $\vec{x}_i \in R^n$ and $y_i \in \{0, 1\}$, $1 \leq i \leq m$, LR defines the class of $\vec{x}_i$ by Eq. (10):

$$y_i' = \begin{cases} 0, & p_i < 0.5 \\ 1, & p_i \geq 0.5 \end{cases} \tag{10}$$

where p_i is calculated by Eq. (11) and σ is the sigmoid function defined by Eq. (12):

$$p_i = \sigma(\vec{w}\vec{x}_i), \tag{11}$$

$$\sigma(a) = \frac{1}{1 + e^{-a}}. \tag{12}$$

The goal of the LR algorithm is to learn weights ($\vec{w}$) that minimize a cost function, such as a cross-entropy function with ridge regression, which reduces the model's complexity and prevents overfitting. The cross-entropy function with ridge regression is given by Eq. (13).

$$J(\vec{w}) = -\sum_{i=1}^{m} y_i \log(p_i) + (1 - y_i)\log(1 - p_i) + \frac{\lambda}{2}\sum_{j=1}^{n} w_j^2, \tag{13}$$

where λ is a regularization parameter. The gradient descent algorithm starts with some initial $\vec{w}$ and repeats the update in Eq. (14) until the termination criteria are satisfied.

$$\vec{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}_{n\times 1} + \alpha \begin{bmatrix} -\vec{x}_1- \\ -\vec{x}_2- \\ \vdots \\ -\vec{x}_m- \end{bmatrix}_{m\times n}^{T} \times \left(\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} - \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_m \end{bmatrix} \right)_{m\times 1} - \lambda \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}_{n\times 1}, \tag{14}$$

where α is the learning rate, which affects how expeditiously the learning model can converge to local minima. If α is too small, slow convergence may occur; if α is too large, gradient descent can overshoot a local minimum. α and λ are important hyper-parameters of LR models.

Tutun et al. [33] applied the evolutionary strategy and simulated annealing to optimize coefficients of LR. Their proposed model was tested on a publicly available diabetes readmission dataset, and the results showed that the proposed model outperformed SVMs, decision trees, NB, and LR.

4. The proposed model and its constituents

4.1. Artificial Bee Colony (ABC) algorithm

The ABC optimization algorithm proposed by Karaboga [9] mimics the intelligent foraging behavior of a honeybee swarm. In the ABC algorithm, there are three types of artificial bees: employed, onlooker, and scout bees. The position of a food source corresponds to a possible solution, and the nectar amount of a food source denotes the quality of the solution. The aim of the algorithm is to find the food source having the maximum nectar amount. The main steps of the algorithm are given in Algorithm 1.

The employed bee phase is responsible for finding better regions around food source positions in memory; employed bees also share the nectar amount and position information of food sources with onlooker bees waiting in the dance area. Each onlooker bee decides on a source to fly to by watching the dance performed by the employed bees. The ABC algorithm simulates the dance and profitable source selection by a stochastic selection scheme such as roulette wheel selection. The stochastic scheme introduces positive feedback, such that if the nectar amount of a food source is high, a larger number of onlooker bees are recruited to the source.

In the greedy selection performed in steps 3 and 7, if the nectar amount of a new solution is higher than that of the current solution, the bee saves the new solution in its memory and forgets the old solution. Otherwise, it keeps the current solution in its memory and increases the counter associated with the current solution by 1. The counters hold the number of exploitations around each source and are used in the scout bee phase to decide whether a food source is exhausted. An exhausted solution is one whose counter exceeds a predetermined number, called the limit. In each cycle of the basic algorithm, at most one employed bee can become a scout bee. If more than one employed bee whose food source has been exhausted exists, the algorithm chooses the source with the highest numbered counter, and this bee is transformed into a scout bee. Exhausted sources are abandoned by the bees, and scout bees search for undiscovered sources to replace them.

Algorithm 1 The ABC Algorithm

```
1: Assign values to the control parameters: the number of food sources ($SN$), the maximum cycle number ($MCN$), and $limit$ value
2: Generate initial food source population using Eq. (15) and evaluate the solutions
       $x_{ij} = x_j^{min} + rand(0,1) \times (x_j^{max} - x_j^{min})$    (15)
   where $x_{ij}$ is the $j^{th}$ parameter of the $i^{th}$ solution; $i \in \{1, 2 \ldots SN\}$ and $j \in \{1, 2 \ldots D\}$, $SN$ and $D$ are the number of food sources and the number of optimization parameters, respectively; $rand(0,1)$ is a uniformly distributed random number within the interval $(0,1)$; and $x_j^{max}$ and $x_j^{min}$ are the upper and lower bounds of the $j^{th}$ parameter, respectively.
3: The employed bee phase applies a local search for each solution by using Eq. (16) and performs greedy selection:
       $v_{ij} = x_{ij} + \varphi_{ij} \times (x_{ij} - x_{kj}),$    (16)
   where $\varphi_{ij}$ is a random number between $[-1,1]$ and $j \in \{1, 2 \ldots D\}$ and $k \in \{1, 2 \ldots SN\}$ are randomly chosen indices, provided that $i \neq k$.
4: Each solution is assigned a probability value proportional to its quality by Eq. (17):
       $p_i = \left(0.9 \times \frac{f_i}{max(f)}\right) + 0.1,$    (17)
   where $f_i$ is the fitness value of the $i^{th}$ solution and $max(f)$ is the maximum fitness value.
5: The onlooker bee phase selects solutions based on their probability values. A local search is performed around each selected solution by using Eq. (16), and greedy selection is applied
6: Memorize the best solution
7: The scout bee phase generates a new solution by using Eq. (15) instead of an exhausted source, if one exists
8: Repeat steps 3–7 until the termination criteria are satisfied
```

4.2. LR classifier based on the ABC algorithm: ABC-LR

Because the gradient descent algorithm used in LR has some limitations, such as requiring derivation calculations and assuming the cost function to be continuous, a successful heuristic, the ABC algorithm, that does not make assumptions about the function and parameter search space is used to train the LR model. In training LR with ABC, the weights of the LR model correspond to the food source locations in the ABC algorithm. Therefore, first, an initial population of weights and bias values are generated by the algorithm. Then, the employed bee, onlooker bee, and scout bee phases search for an optimum weight set ($\vec{w}_i$) and bias value that minimize the sum squared error at the output of the model.

In the ABC algorithm, a food source position vector corresponding to the weights and bias value is initialized using Eq. (18).

$$w_{ij} = w_j^{min} + rand(0,1) \times (w_j^{max} - w_j^{min}), \tag{18}$$

where $\vec{w}_i$ is the ith food source position, $i = 1 \ldots SN$, SN is the number of food sources, $rand(0,1)$ is a uniformly distributed random number within the interval $(0,1)$, w_j^{max} and w_j^{min} refer to the upper and lower bounds for jth parameter, respectively, and $j = 1 \ldots n$ and n is the number of parameters to be optimized.

Each weight vector set is applied to the LR model, and an output is calculated using the weights and bias value in the solution vector. The fitness of a solution (Eq. (19)) is calculated using the error (Eq. (20)) between the desired output and the output calculated by the LR model. This fitness function and selection operator guides the algorithm in the search space to the better regions.

$$f_i = \frac{1}{1 + E_i} \tag{19}$$

$$E_i = \sum_{j=1}^{m} (p_j^i - y_j^i)^2 \tag{20}$$

Once the initial population is evaluated, the algorithm starts to iterate the employed bee, onlooker bee, and scout bee phases until the termination criteria are satisfied. In the employed bee phase, a local search is conducted around each solution in memory and a new solution is obtained by Eq. (21).

$$v_{ij} = w_{ij} + \varphi_{ij} \times (w_{ij} - w_{kj}), \tag{21}$$

where φ_{ij} is a random number drawn from a uniform distribution within range $[-1,1]$, j is a random dimension index in the interval $[1,n]$, and k is a randomly chosen neighbor such that $i \neq k$.

In the standard version of the ABC algorithm, by Eq. (21), a modification is performed on just one randomly chosen parameter j to generate a new solution. In this study, to improve the convergence rate, a modified local search operator is used. By the operator given in Eq. (22), each parameter can be changed if a random number drawn from a uniform distribution within the range $[0,1]$ is less than MR, which is a predefined control parameter.

$$v_{ij} = \begin{cases} w_{ij} + \varphi_{ij} \times (w_{ij} - w_{kj}), & R_{ij} < MR \\ w_{ij}, & otherwise \end{cases} \tag{22}$$

where R_{ij} is a random number drawn from a uniform distribution within the range [0,1].

Once a solution is generated, Eq. (23) determines whether it violates the parameter boundaries.

$$v_{ij} = \begin{cases} w_j^{max}, & v_{ij} > w_j^{max} \\ w_j^{min}, & v_{ij} < w_j^{min} \\ v_{ij}, & otherwise \end{cases} \tag{23}$$

The new solution is evaluated in the fitness function, and a greedy selection is applied between the current solution and the new one.

Each solution is assigned a probability value by Eq. (17) proportional to its quality or fitness. An onlooker bee chooses a food source based on the probability value and generates a new solution by a local search using Eq. (22) as each employed bee does. Greedy selection is applied between the current solution and new solution.

As in the standard ABC algorithm, if an exhausted source exists, the scout bee phase abandons it and creates a new food source position (solution) using Eq. (18).

5. Experiments

5.1. Datasets

In this research, three publicly available datasets were used to examine the performance of the algorithms for filtering spam emails. The first is called the CSDMC2010 spam corpus, which is one of the datasets for the data mining competition associated with ICONIP 2010. This dataset contains a total of 4327 English

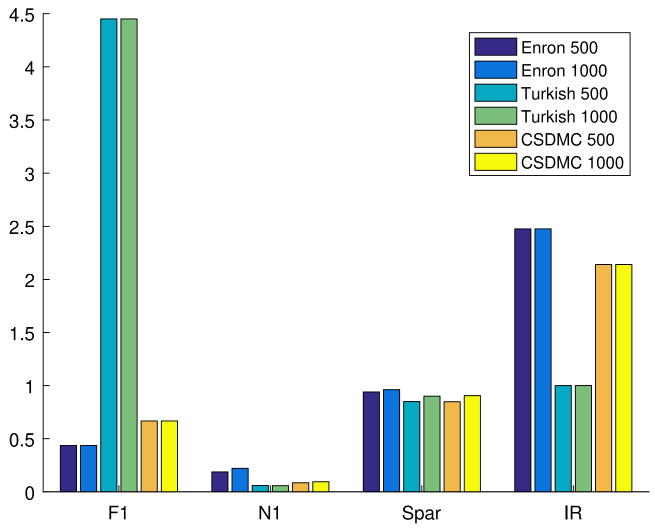

Fig. 3. Dataset evaluation criteria.

email messages, of which 2949 are legitimate emails (68.15%) and 1378 (31.85%) are spam emails.

The second dataset, called the TurkishEmail dataset, consists of Turkish emails; it contains 400 spam emails (50%) and 400 legitimate emails (50%), for a total of 800 emails [34].

The third dataset is the Enron-1 dataset [15], which is a popular corpus used as a benchmark in many studies. It consists of 5172 emails, of which are 3672 legitimate emails (71%) and 1500 (29%) are spam emails. The emails are nonencoded and in their original forms. Legitimate emails belong to just one person (i.e. they are personalized) while spam emails were collected by the third author of the paper (Paliouras) [15].

The CSDMC2010, TurkishEmail, and Enron datasets comprise 82 148, 25 650, and 47 939 distinct terms, respectively.

We first examined the datasets in terms of their complexity, sparsity, and imbalance ratio. To measure the datasets' complexity, we applied some of the methods proposed by Tin and Basu [35]. The first complexity measure is Fisher's discriminant ratio (F1), which computes the overlaps between the features in different classes. The second method, called N1, measures the separability of classes' distributions. High F1 values indicate that there is at least one feature with little overlap between the classes, whereas low F1 values indicate more complex problems, where no individual feature separates the classes. Therefore, the F1 values of the datasets show that the TurkishEmail dataset is a more linear problem, which includes at least one discriminative feature, than the other two datasets; whereas, the Enron dataset represents a nonlinear problem (Fig. 3). Higher values of N1 demonstrate that more complex boundaries are needed to discriminate between classes and that there is more overlap between the classes. According to Fig. 3, the Enron dataset needs more complex boundaries to separate the classes compared to the other two datasets. The datasets are also evaluated in terms of their sparsity and imbalance ratio. The imbalance ratio is calculated by dividing the number of legitimate emails by the number of spam mails. The imbalance ratios of the Enron, TurkishEmail, and CSDMC2010 datasets are 2.47, 1, and 2.14, respectively. The Enron and CSDMC2010 datasets are imbalanced, whereas the TurkishEmail dataset is balanced. The sparsity of the Enron, TurkishEmail, and CSDMC2010 datasets are 96.09%, 90.02%, and 90.48%, respectively, when the feature vector size equals 1000. The sparsity percentages indicate that the datasets are sparse.

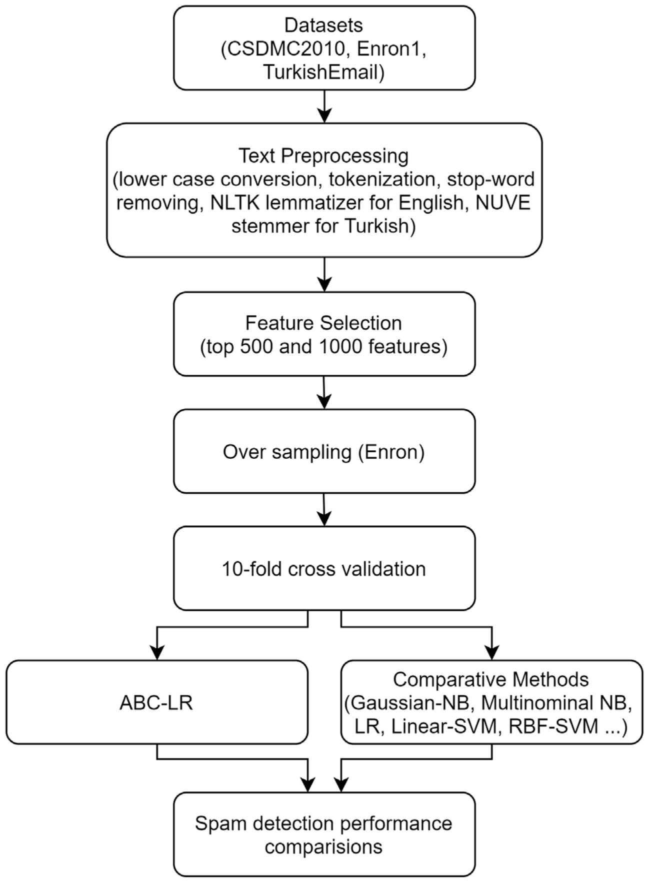
Datasets (CSDMC2010, Enron1, TurkishEmail)
Text Preprocessing (lower case conversion, tokenization, stop-word removing, NLTK lemmatizer for English, NUVE stemmer for Turkish)
Feature Selection (top 500 and 1000 features)
Over sampling (Enron)
10-fold cross validation
ABC-LR
Comparative Methods (Gaussian-NB, Multinominal NB, LR, Linear-SVM, RBF-SVM ...)
Spam detection performance comparisions

Fig. 4. Spam detection model.

5.2. Preprocessing for data representation

In the spam mail classification problem, we are given a set of emails $D = \{d_1, d_2, \ldots, d_m\}$ and two classes $C = \{c_{spam}, c_{legitimate}\}$. The problem is to assign an email $d_i \in D$ to one of the two classes. The main steps of the research methodology are given in Fig. 4.

In order to process a given email in the English and Turkish datasets, first, the body and subject of the email are extracted, and the html tags are removed because they are noninformative. All the letters are converted to lowercase and tokenization is performed to divide the strings into lists of substrings. Stop words and punctuation are removed from the substrings, except for exclamation marks, which are not discriminative expressions in spam email. Since the structures and word morphologies in Turkish and English languages have significant differences, two different stemming libraries are used. For the English dataset, to remove morphological affixes from words and leave only the word stem, the NLTK stemming library [36] is applied to the words. The preprocessing steps in our study can be summarized as in Algorithm 2 [37].

Once the preprocessing steps are performed, machine-learning algorithms are applied to the data, which are expressed as a reduced vector representation, and the classifier maps documents to classes, as shown in Eq. (27).

$$\phi(\vec{d}'_i, c_j) = \begin{cases} 1, & if \quad \vec{d}'_i \in c_j \\ 0, & otherwise \end{cases} \tag{27}$$

5.3. Feature selection approach

Training a classifier by using all the terms requires a high computation time and source. Therefore, to eliminate redundant

Algorithm 2 Data preprocessing steps

```
1: All characters are converted to lowercase and all the words and symbols are tokenized to meaningful parts (tokenization).
2: All nontext items, such as punctuation, html tags, and spaces, are removed to determine candidate words (pruning).
3: The derived words are reduced to their morphological bases by removing plurals, prefixes, and suffixes (stemming).
4: Noninformative common stop-words are eliminated and the bag of words (BOW) model is obtained independently of the grammar and order of words. The BOW model constitutes a vocabulary (term set) which consists of all the unique terms in all documents.
       $B = \{t_1, t_2, ..., t_{|B|}\},$   (24)
   where $B$ and $|B|$ are the term set and the total number of unique terms occurring in all documents, respectively.
5: After the term set is created, each document $d_i \in D$ is represented by a vector $\vec{d_i} = [w_1, w_2, ..., w_{|B|}]$, where $w_1, w_2, ..., w_{|B|}$ are the weights that correspond to the terms $t_1, t_2, ..., t_{|B|}$. The weights are computed using $tf - idf$.
       $tf - idf(t_j, d_i) =< w_j, d_i >= tf(t_j, d_i) \times log \frac{|D|}{|D_{t_j}|},$   (25)
   where $tf(t_j, d_i)$ is the total number of terms $t_j$ that exist in document $d_i$, $|D|$ refers to the total number of documents, and $|D_{t_j}|$ refers to the total number of documents that contain term $t_j$
6: As the number of words increases, the vector space dimension increases, which makes the computational complexity high. Hence, feature selection or space transformation approaches are used to reduce space dimensionality. Once the selection of the more informative terms is completed, a document $d_i$ is represented as a feature vector $\vec{d'_i}$ that consists of the corresponding weights for these selected terms.
       $\vec{d'_i} = [w_{1'}, w_{2'}, ..., w_{|B'|}], |B'| << |B|$   (26)
```

features and reduce computational complexity, efficient feature selection methods are used to select a subset of terms instead of whole words. The feature selection methods determine a set of critical terms that form a feature set ($S = t_1, \ldots, t_n$). In our experiments, each email is represented by a feature vector $\vec{x} = [x_1, x_2, \ldots, x_n]$, where $x_1, x_2, \ldots, x_n$ are the corresponding values to the terms, and n stands for the feature vector size. Additionally, feature selection improves the success of a classifier by excluding non-discriminative words that exist in almost all classes. In this study, a $tf - idf$)-based feature selection method is used. Previous studies [38,39] have also used $tf - idf$-based feature selection methods as an effective feature selection technique with considerable success. $tf - idf$ (Eq. (25)) shows how important a term is in a document. After all the $tf - idf$ vectors are computed, the vectors are normalized by the Euclidean ($L2$) norm, which is a well-known normalization technique, as in Eq. (28) [40]. As the probability of a term appearing in the document gets higher, the term seems more important than it is in long documents. The aim of normalization is to prevent this bias in long documents by considering document length. Amayri and Bouguila [22] applied normalization methods to $tf - idf$ vectors and mentioned that using normalization can prevent sparse data attacks and that applying the L2-norm produced better results compared to applying the L1-norm.

$$v^i_{norm} = \frac{v^i}{v^i_1 + v^i_2 + \cdots + v^i_n} \quad (i = 1...N) \tag{28}$$

The total weight of a term is the sum of the corresponding normalized $tf - idf$ weights of the term in all documents. Once the total weight of each term is computed, the weights are sorted in descending order. The terms up to feature vector size n with the highest total weight are selected to form the feature set. Finally, each document can be represented by a feature vector that consists of the normalized $tf - idf$ weights of these terms.

Table 1
Classification accuracies of a multinomial NB classifier on the TurkishEmail dataset, FVS: Feature Vector Size.

FVS	α (smoothing parameter)				
	$1.0e-10$	0.001	0.01	0.1	1
500	96.88%	96.75%	96.75%	96.50%	95.63%
1000	97.62%	97.62%	97.62%	97.38%	97.12%

6. Results and discussion

In the first part of the experiments, the results of multinomial and Gaussian NB classifiers were obtained, and the sensitivity of multinomial NB against the smoothing parameters was analyzed. In the second part of the experiments, linear and radial basis SVMs were applied, and their sensitivity against the box constraint was investigated. In the third part of the experiments, an LR classifier trained by the gradient descent method and the proposed model were examined. The effects of learning rate, regularization parameters, feature vector size, and the control parameters of ABC on classification accuracy were presented. In the last part of the experiments, Gaussian NB, Multinomial NB, linear SVM, radial basis SVM, LR, and the proposed model were compared in terms of their classification accuracy on the TurkishEmail and CSDMC2010 datasets. In addition, the proposed model's classification performance on the Enron dataset was compared with that of other methods from previous studies. In all experiments, tenfold cross-validation was performed to eliminate the effect of the order and splitting scheme on the data. In the tenfold cross-validation method, the dataset is divided into ten parts; one part is used for testing while the other parts are used for training. Thus, ten classification results are obtained, and the average of these results is calculated as the final classification accuracy of that experiment.

6.1. Classification by NB classifiers

6.1.1. Classification by multinomial NB classifier

A multinomial NB classifier was trained with two different feature vector sizes {500, 1000} and with the following parameters: $\alpha = \{1.0e - 10, 0.001, 0.01, 0.1, 1\}$. Ten experiments were performed for each dataset using the scikit-learn library [41]. In the experiments, normalized tf-idf valued feature vectors were used. Although feature vectors generally consist of term counts in multinomial NB, tf-idf valued feature vectors are also known to yield successful results in practice [40]. Tables 1 and 2 show the classification accuracies of the multinomial NB classifier on the TurkishEmail and CSDMC2010 datasets, respectively. As seen from Tables 1 and 2, the best results are obtained when α is very close to 0.

Table 2
Classification accuracies of a multinomial NB classifier on the CSDMC2010 dataset.

FVS	α (smoothing parameter)				
	1.0e−10	0.001	0.01	0.1	1
500	90.93%	89.79%	89.51%	89.30%	88.89%
1000	94.55%	93.79%	93.50%	93.18%	92.02%

Table 3
Classification accuracies of the Gaussian NB classifier.

FVS	TurkishEmail dataset	CSDMC2010 dataset
500	95.38%	93.71%
1000	96.63%	95.92%

6.1.2. Classification by a Gaussian NB classifier

A Gaussian NB classifier was trained with two feature vector sizes {500, 1000}, and two experiments were performed for each dataset using the scikit-learn library [41]. Table 3 shows the classification accuracies of the Gaussian NB classifier on the TurkishEmail and CSDMC2010 datasets. Because the TurkishEmail dataset has fewer distinct words compared to the CSDMC2010 dataset, the difference in classification accuracy with respect to feature vector size is less in TurkishEmail than in the CSDMC2010 dataset.

6.2. Classification by SVM classifiers

6.2.1. Classification by linear SVM classifier

A linear SVM classifier was trained with two feature vector sizes {500, 1000} and with the following box constraint parameters: $C = \{2^{-3}, 2^{-2}, 2^{-1}, 2^{0}, 2^{1}, 2^{2}, 2^{3}, 2^{4}, 2^{5}\}$. We performed a grid search technique while determining the parameter ranges [42]. For each dataset, 18 experiments were performed using the LibSVM library [43].

Tables 4 and 5 list the classification accuracies of a linear SVM classifier on the TurkishEmail and CSDMC2010 datasets, respectively. On the TurkishEmail dataset, the best result is achieved when C is 2^4 if the feature vector size is 500, and when C is 2^5 if the feature vector size is 1000. On the CSDMC2010 dataset, the best result is achieved when C is 2^2 if the feature vector size is 500, and when C is 2^4 if the feature vector size is 1000. The worst results are obtained when C is 2^{-3} for both the TurkishEmail and CSDMC2010 datasets. As seen from the results, the algorithm performance is sensitive to the value of C.

6.2.2. Classification by a radial basis SVM classifier

An SVM classifier with RBF kernel ($K(x_i, x_j) = exp(-\gamma\|x_i - x_j\|^2), \gamma > 0$) was trained with two feature vector sizes {500, 1000} and with the following parameters: $C = \{2^{-3}, 2^{-2}, \dots, 2^{4}, 2^{5}\}$ and $\gamma = \{2^{-15}, 2^{-13}, \dots, 2^{3}, 2^{5}\}$. We used a grid search technique to decide which parameters were suitable [42]. For each dataset and feature vector size, 99 experiments were carried out using the LibSVM library [43]. The experimental results are shown in Tables 6, 7, 8, and 9. The best result was achieved when $C = 2^5$, $\gamma = 2^1$, and FVS = 1000, which produced a 98.75% success rate on the TurkishEmail dataset. On the CSDMC2010 dataset, the best result was achieved when $C = 2^3$, $\gamma = 2^{-1}$, and FVS = 1000. The experimental results also demonstrated that the SVM classifier with RBF kernel achieved slightly better classification accuracies for feature vector sizes of 500 and 1000 on the CSDMC2010 dataset, whereas the SVM classifier with a linear kernel achieved slightly better classification accuracies for feature vector sizes of 500 and 1000 on the TurkishEmail dataset.

6.3. Classification by LR classifier

6.3.1. Classification by LR trained by gradient descent algorithm

In the traditional LR model, the gradient descent algorithm is used to minimize the error function defined by Eq. (14). The LR classifier was trained with two feature vector sizes {500,1000}, three learning rates α {0.001, 0.01, 0.1}, and four regularization parameters λ {0, 0.001, 0.01, 0.1}. We performed an exhaustive grid method for tuning purpose. Twenty-four configurations were generated for each dataset, and each configuration was repeated 30 times with different random seeds. In Tables 10 and 11, the best, worst, median, mean, and standard deviations of classification accuracies obtained from 30 runs of each configuration are reported for the TurkishEmail and CSDMC2010 datasets, respectively. Fig. 5 summarizes all runs using box plot graphs.

In Tables 10 and 11 and Fig. 5, on both the TurkishEmail and CSDMC2010 datasets, when learning rate α and regularization parameter λ are 0.1, the algorithm produces unstable results. The most robust results are achieved when the learning rate is 0.01, and smaller regularization parameter values {0, 0.001} improve performance. As seen from the box plots in Fig. 5, 500 features can represent the data well enough if we consider computational complexity in the case when the feature vector size is 1000.

6.3.2. Classification by the proposed model: LR trained by the ABC algorithm

Because the ABC algorithm has some control parameters, we aimed to recommend suitable parameter ranges that yield optimal performance for this type of problem [44,45]. The appropriate values extracted were used in the subsequent part of the study.

The ABC algorithm has several control parameters, such as the number of food sources, the maximum cycle number, the *limit* to abandon a food source, and modification rate. We performed an exhaustive grid method for tuning purposes. The values {40, 60, 80} were assigned to the *SN* parameter in these experiments. To conduct a fair comparison, the evaluation number was fixed

Table 4
Classification accuracies of a linear SVM classifier on the TurkishEmail dataset.

FVS	C								
	2^{-3}	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}	2^{3}	2^{4}	2^{5}
500	93.00%	94.25%	96.37%	97.50%	98.00%	97.88%	98.13%	98.63%	98.50%
1000	93.75%	95.38%	96.37%	97.75%	98.00%	98.38%	98.50%	98.75%	98.88%

Table 5
Classification accuracies of a linear SVM classifier on the CSDMC2010 dataset.

FVS	C								
	2^{-3}	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}	2^{3}	2^{4}	2^{5}
500	92.30%	95.36%	96.64%	97.49%	97.87%	98.10%	97.96%	98.07%	97.82%
1000	94.08%	96.08%	97.33%	97.98%	97.96%	98.24%	98.35%	98.42%	98.19%

Table 6
Classification accuracies of the SVM classifier with RBF kernel on the TurkishEmail dataset for FVS = 500.

γ	C								
	2^{-3}	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}	2^{3}	2^{4}	2^{5}
2^{-15}	53.75%	53.75%	53.75%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%
2^{-13}	53.75%	53.75%	53.75%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%
2^{-11}	53.75%	53.75%	53.75%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%
2^{-9}	53.75%	53.75%	53.75%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%
2^{-7}	53.75%	53.75%	53.75%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%
2^{-5}	54.00%	54.00%	54.00%	59.37%	61.37%	61.37%	61.37%	61.37%	61.37%
2^{-3}	55.87%	55.87%	67.37%	81.87%	82.25%	82.25%	82.25%	82.25%	82.25%
2^{-1}	94.50%	95.87%	97.25%	97.87%	98.12%	98.12%	98.25%	98.37%	98.37%
2^{1}	90.12%	92.87%	94.00%	96.00%	97.25%	98.00%	98.00%	98.25%	98.62%
2^{3}	87.50%	87.50%	87.87%	88.87%	90.62%	93.00%	94.25%	96.37%	97.50%
2^{5}	87.50%	87.50%	87.50%	87.50%	87.50%	87.50%	87.87%	88.87%	90.62%

Table 7
Classification accuracies of the SVM classifier with RBF kernel on the TurkishEmail dataset for FVS = 1000.

γ	C								
	2^{-3}	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}	2^{3}	2^{4}	2^{5}
2^{-15}	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%
2^{-13}	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%
2^{-11}	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%
2^{-9}	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%
2^{-7}	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%	54.00%
2^{-5}	54.00%	54.00%	54.00%	56.62%	57.75%	57.75%	57.75%	57.75%	57.75%
2^{-3}	54.00%	54.00%	61.75%	79.50%	80.00%	80.00%	80.00%	80.00%	80.00%
2^{-1}	89.62%	94.00%	96.00%	96.50%	97.25%	97.25%	97.25%	97.25%	97.25%
2^{1}	90.25%	93.62%	95.00%	96.00%	97.62%	98.00%	98.25%	98.50%	98.75%
2^{3}	87.75%	87.75%	88.00%	89.25%	91.12%	93.62%	95.37%	96.37%	97.75%
2^{5}	87.75%	87.75%	87.75%	87.75%	87.75%	87.75%	88.12%	89.50%	91.25%

Table 8
Classification accuracies of the SVM classifier with RBF kernel on the CSDMC2010 Dataset for FVS = 500.

γ	C								
	2^{-3}	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}	2^{3}	2^{4}	2^{5}
2^{-15}	68.21%	68.37%	69.04%	71.29%	71.29%	71.29%	71.29%	71.29%	71.29%
2^{-13}	68.21%	68.37%	69.04%	71.29%	71.29%	71.29%	71.29%	71.29%	71.29%
2^{-11}	68.21%	68.37%	69.04%	71.53%	71.53%	71.53%	71.53%	71.53%	71.53%
2^{-9}	68.21%	68.37%	69.21%	72.06%	72.11%	72.11%	72.11%	72.11%	72.11%
2^{-7}	68.21%	68.53%	69.41%	72.38%	72.41%	72.41%	72.41%	72.41%	72.41%
2^{-5}	68.21%	68.56%	69.97%	73.61%	73.94%	73.94%	73.94%	73.94%	73.94%
2^{-3}	69.21%	70.32%	73.59%	77.63%	78.95%	78.95%	78.93%	78.93%	78.93%
2^{-1}	96.96%	97.56%	98.02%	98.28%	98.44%	98.51%	98.56%	98.58%	98.58%
2^{1}	85.82%	92.13%	95.05%	96.84%	97.54%	98.00%	98.30%	98.12%	98.16%
2^{3}	68.21%	68.21%	74.17%	78.86%	86.89%	92.29%	95.40%	96.65%	97.47%
2^{5}	68.21%	68.21%	68.21%	68.21%	68.21%	68.21%	74.33%	78.90%	86.91%

Table 9
Classification accuracies of the SVM classifier with RBF kernel on the CSDMC2010 Dataset for FVS = 1000.

γ	C								
	2^{-3}	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}	2^{3}	2^{4}	2^{5}
2^{-15}	68.21%	68.37%	69.04%	71.29%	71.29%	71.29%	71.29%	71.29%	71.29%
2^{-13}	68.21%	68.37%	69.04%	71.29%	71.29%	71.29%	71.29%	71.29%	71.29%
2^{-11}	68.21%	68.37%	69.04%	71.48%	71.48%	71.48%	71.48%	71.48%	71.48%
2^{-9}	68.21%	68.37%	69.11%	71.83%	71.92%	71.92%	71.92%	71.92%	71.92%
2^{-7}	68.21%	68.53%	69.39%	72.11%	72.20%	72.20%	72.20%	72.20%	72.20%
2^{-5}	68.21%	68.53%	69.79%	73.15%	73.45%	73.45%	73.45%	73.45%	73.45%
2^{-3}	68.81%	69.97%	72.94%	76.58%	77.44%	77.44%	77.44%	77.44%	77.44%
2^{-1}	95.77%	97.51%	98.02%	98.44%	98.51%	98.58%	98.63%	98.58%	98.58%
2^{1}	87.05%	93.45%	95.77%	97.37%	98.00%	98.07%	98.28%	98.35%	98.37%
2^{3}	68.21%	68.21%	74.68%	80.53%	88.05%	94.08%	96.05%	97.33%	98.02%
2^{5}	68.21%	68.21%	68.21%	68.21%	68.21%	68.21%	74.82%	80.62%	88.07%

to the same value (160 000) for all experiments. Therefore, the maximum cycle number was set to {2000, 1333, 1000} when *SN* was {40, 60, 80}, respectively. For the *MR* control parameter, values of {0.05, 0.08, 0.1, 0.2} were investigated. The experiments were repeated for the values {100, 200} of the *limit* parameter. The feature vector size was analyzed for the values of {500, 1000}. Each of 48 configurations was repeated 30 times for both the TurkishEmail and CSDMC2010 datasets. The parameter range for $[w_j^{min}, w_j^{max}]$ was $[-8, 8]$.

Each experiment was repeated 30 times, and the best, worst, mean, and standard deviation values of 30 runs of each experiment can be seen in Tables 12–13. Figs. 6–8 show the effects of

Table 10
LR classification statistics on the TurkishEmail dataset.

α	λ	Feature vector size = 500					Feature vector size = 1000				
		Best	Worst	Median	Mean	Std.	Best	Worst	Median	Mean	Std.
0.001	0	95.50%	94.50%	95.00%	94.98%	0.31	95.88%	94.63%	95.25%	95.22%	0.31
	0.001	95.50%	94.25%	95.00%	94.97%	0.34	96.00%	94.25%	95.06%	95.13%	0.37
	0.01	95.88%	94.25%	95.00%	94.99%	0.35	95.75%	94.62%	95.25%	95.25%	0.28
	0.1	95.63%	94.37%	95.00%	95.01%	0.30	95.75%	94.38%	95.25%	95.15%	0.38
0.01	0	98.00%	97.50%	97.75%	97.78%	0.12	98.00%	97.25%	97.75%	97.72%	0.17
	0.001	98.00%	97.50%	97.75%	97.76%	0.12	98.13%	97.50%	97.75%	97.75%	0.15
	0.01	98.00%	97.50%	97.75%	97.72%	0.11	98.00%	97.37%	97.75%	97.73%	0.16
	0.1	97.75%	97.50%	97.63%	97.64%	0.07	97.75%	97.37%	97.63%	97.58%	0.10
0.1	0	98.38%	98.00%	98.25%	98.24%	0.09	98.75%	98.25%	98.50%	98.48%	0.12
	0.001	98.50%	98.13%	98.25%	98.29%	0.08	98.63%	98.38%	98.50%	98.53%	0.08
	0.01	98.50%	98.25%	98.25%	98.31%	0.07	98.50%	98.25%	98.38%	98.36%	0.10
	0.1	97.63%	90.25%	96.38%	95.32%	2.32	97.88%	91.38%	96.88%	95.82%	2.14

Table 11
LR classification statistics on the CSDMC2010 dataset.

α	λ	Feature vector size = 500					Feature vector size = 1000				
		Best	Worst	Median	Mean	Std.	Best	Worst	Median	Mean	Std.
0.001	0	96.45%	95.87%	96.09%	96.10%	0.13	96.75%	96.22%	96.48%	96.49%	0.14
	0.001	96.38%	95.78%	96.11%	96.13%	0.16	96.80%	96.22%	96.51%	96.52%	0.16
	0.01	96.36%	95.80%	96.05%	96.06%	0.14	96.89%	96.08%	96.50%	96.50%	0.17
	0.1	96.31%	95.80%	96.03%	96.06%	0.12	96.82%	96.29%	96.50%	96.53%	0.14
0.01	0	98.14%	97.87%	98.03%	98.02%	0.06	98.40%	98.14%	98.26%	98.27%	0.06
	0.001	98.19%	97.91%	98.03%	98.04%	0.07	98.38%	98.10%	98.24%	98.24%	0.07
	0.01	98.12%	97.84%	97.99%	97.99%	0.07	98.38%	98.17%	98.26%	98.27%	0.05
	0.1	97.98%	97.80%	97.89%	97.89%	0.05	98.31%	98.07%	98.21%	98.21%	0.05
0.1	0	98.26%	98.10%	98.17%	98.17%	0.05	98.56%	98.42%	98.49%	98.49%	0.03
	0.001	98.17%	95.20%	98.03%	97.94%	0.53	98.56%	98.40%	98.47%	98.47%	0.04
	0.01	98.03%	91.65%	96.10%	95.87%	1.96	98.47%	93.09%	98.27%	97.46%	1.42
	0.1	85.15%	73.85%	78.93%	79.32%	3.23	91.21%	66.75%	81.24%	80.87%	6.17

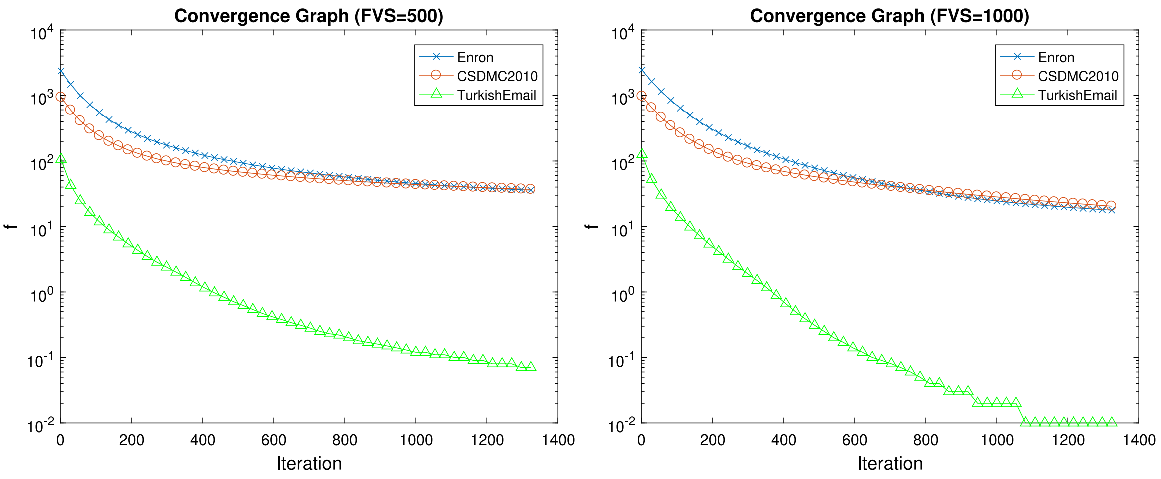

Fig. 5. Box plot graphs of LR algorithm classification results obtained for different learning rates, regularization parameters, and feature vector sizes on the TurkishEmail and CSDMC2010 datasets.

the parameters (*SN*, *MR*, and *limit*) and of the number of features {500, 1000} on the TurkishEmail and CSDMC2010 datasets.

From Tables 12–13 and Figs. 6–8, we see that the best values are obtained as the feature vector size increases, as we expected. However, on the CSDMC2010 dataset, the feature vector size has high impact compared to that on the TurkishEmail dataset. This difference may arise from the fact that the CSDMC2010 dataset has more distinct words than the TurkishEmail dataset. Because the number of evaluations was fixed to the same value for all experiments, various *SN* values produce similar results. However, as seen in Figs. 6 and 8, when *SN* is set to 60, the algorithm achieved slightly better and more robust results on both the TurkishEmail and CSDMC2010 datasets. On the TurkishEmail dataset, the best results were obtained with $MR = 0.2$ and $MR = 0.1$ when the feature vector size was 500 and 1000, respectively, while the range $[0.05 - 0.1]$ is more stable. The value 0.05 for *MR* produces more efficient and robust results on the CSDMC2010 dataset. As seen from Figs. 6 and 8, the algorithm produced similar results when the *limit* value was 100 and 200.

To compare the classification performance of ABC-LR with previous studies, it was applied to the Enron 1 dataset and trained with three different $SN = \{40, 60, 80\}$, two different $MR = \{0.05, 0.1\}$, and $limit = 100$. The maximum iteration number was set to {2000, 1333, 1000} to fix the evaluation number as 160 000.

Table 12
LR classification statistics trained by the ABC algorithm on the TurkishEmail dataset.

SN	*MR*	*limit*	Feature vector size = 500					Feature vector size = 1000				
			Best	Worst	Median	Mean	Std.	Best	Worst	Median	Mean	Std.
40	0.05	100	98.87%	98.38%	98.62%	98.61%	0.12	99.00%	98.38%	98.75%	98.71%	0.16
		200	98.87%	98.37%	98.62%	98.60%	0.17	99.13%	98.38%	98.88%	98.82%	0.20
	0.08	100	99.00%	98.38%	98.62%	98.62%	0.17	99.00%	98.38%	98.75%	98.73%	0.17
		200	99.00%	98.38%	98.62%	98.62%	0.14	99.00%	98.25%	98.75%	98.68%	0.19
	0.1	100	99.00%	98.25%	98.62%	98.58%	0.15	99.25%	98.25%	98.63%	98.64%	0.20
		200	98.88%	98.00%	98.62%	98.58%	0.20	99.13%	98.50%	98.81%	98.76%	0.20
	0.2	100	99.13%	98.37%	98.62%	98.61%	0.20	99.00%	98.00%	98.63%	98.59%	0.22
		200	98.88%	98.13%	98.63%	98.57%	0.21	99.00%	98.13%	98.56%	98.56%	0.25
60	0.05	100	98.87%	98.25%	98.63%	98.61%	0.19	99.00%	98.50%	98.88%	98.82%	0.13
		200	98.87%	98.13%	98.62%	98.59%	0.18	99.00%	98.50%	98.75%	98.77%	0.16
	0.08	100	98.87%	98.37%	98.62%	98.60%	0.13	99.13%	98.25%	98.75%	98.73%	0.19
		200	98.75%	98.38%	98.63%	98.60%	0.11	99.13%	98.50%	98.75%	98.79%	0.14
	0.1	100	98.88%	98.38%	98.63%	98.64%	0.16	99.00%	98.38%	98.75%	98.75%	0.16
		200	98.87%	98.38%	98.63%	98.64%	0.12	99.00%	98.38%	98.75%	98.76%	0.18
	0.2	100	98.75%	98.13%	98.50%	98.48%	0.19	99.13%	98.13%	98.63%	98.64%	0.22
		200	98.87%	98.25%	98.63%	98.59%	0.16	99.13%	98.25%	98.69%	98.67%	0.21
80	0.05	100	98.87%	98.25%	98.50%	98.54%	0.15	99.00%	98.50%	98.88%	98.82%	0.13
		200	99.00%	98.25%	98.50%	98.57%	0.19	99.00%	98.50%	98.75%	98.78%	0.14
	0.08	100	98.87%	98.25%	98.62%	98.61%	0.15	99.13%	98.63%	98.88%	98.81%	0.13
		200	98.88%	98.38%	98.50%	98.56%	0.14	99.00%	98.50%	98.88%	98.83%	0.14
	0.1	100	98.75%	98.25%	98.63%	98.60%	0.13	99.13%	98.38%	98.88%	98.78%	0.17
		200	98.88%	98.25%	98.50%	98.56%	0.15	99.00%	98.38%	98.75%	98.77%	0.16
	0.2	100	99.13%	98.00%	98.50%	98.52%	0.23	99.00%	98.38%	98.75%	98.67%	0.16
		200	99.00%	98.38%	98.50%	98.60%	0.16	99.00%	98.25%	98.75%	98.69%	0.20

Table 13
LR classification statistics trained by the ABC algorithm on the CSDMC2010 dataset.

SN	*MR*	*limit*	Feature vector size = 500					Feature vector size = 1000				
			Best	Worst	Median	Mean	Std.	Best	Worst	Median	Mean	Std.
40	0.05	100	98.19%	97.80%	98.03%	98.03%	0.11	98.56%	98.19%	98.39%	98.39%	0.09
		200	98.31%	97.77%	98.03%	98.02%	0.12	98.65%	98.21%	98.45%	98.43%	0.10
	0.08	100	98.17%	97.87%	97.99%	98.00%	0.07	98.65%	98.14%	98.30%	98.32%	0.11
		200	98.19%	97.82%	98.03%	98.02%	0.09	98.49%	98.10%	98.34%	98.33%	0.11
	0.1	100	98.19%	97.73%	97.93%	97.94%	0.11	98.52%	98.03%	98.27%	98.26%	0.13
		200	98.17%	97.75%	97.94%	97.95%	0.10	98.54%	98.03%	98.24%	98.25%	0.12
	0.2	100	98.05%	97.45%	97.75%	97.74%	0.14	98.35%	97.77%	98.04%	98.04%	0.14
		200	98.00%	97.61%	97.75%	97.76%	0.10	98.33%	97.73%	98.06%	98.04%	0.16
60	0.05	100	98.35%	97.96%	98.09%	98.10%	0.10	98.61%	98.24%	98.45%	98.42%	0.10
		200	98.21%	97.91%	98.10%	98.08%	0.07	98.70%	98.24%	98.45%	98.43%	0.11
	0.08	100	98.21%	97.82%	98.07%	98.04%	0.10	98.54%	98.14%	98.41%	98.38%	0.11
		200	98.26%	97.91%	98.06%	98.09%	0.08	98.61%	98.19%	98.37%	98.37%	0.09
	0.1	100	98.19%	97.82%	97.99%	98.01%	0.10	98.56%	98.14%	98.31%	98.31%	0.11
		200	98.26%	97.87%	98.05%	98.04%	0.11	98.58%	98.12%	98.32%	98.34%	0.12
	0.2	100	98.17%	97.56%	97.85%	97.84%	0.14	98.38%	97.82%	98.11%	98.11%	0.15
		200	98.12%	97.47%	97.82%	97.82%	0.15	98.38%	97.66%	98.12%	98.10%	0.15
80	0.05	100	98.17%	97.75%	98.03%	98.04%	0.09	98.58%	98.21%	98.33%	98.34%	0.09
		200	98.17%	97.87%	98.02%	98.01%	0.08	98.49%	98.12%	98.35%	98.33%	0.09
	0.08	100	98.24%	97.80%	98.03%	98.02%	0.11	98.56%	98.14%	98.38%	98.37%	0.11
		200	98.24%	97.70%	98.04%	98.03%	0.12	98.56%	98.10%	98.39%	98.36%	0.11
	0.1	100	98.28%	97.82%	98.04%	98.01%	0.13	98.54%	98.19%	98.34%	98.33%	0.09
		200	98.21%	97.84%	97.99%	97.99%	0.10	98.54%	98.12%	98.31%	98.32%	0.11
	0.2	100	98.07%	97.52%	97.83%	97.83%	0.15	98.31%	97.87%	98.10%	98.09%	0.11
		200	98.07%	97.63%	97.88%	97.88%	0.11	98.31%	97.73%	98.07%	98.06%	0.14

The parameter ranges for w_j^{min} and w_j^{max} were set to -64 and 64, respectively. A total of 12 experiments were performed. The best, worst, median, mean, and standard deviation values of each experiment can be seen in Table 14.

The convergence graphs of the ABC algorithm in the training phase of the LR method on the Enron, CSDMC2010, and Turkish Email data sets are shown in Fig. 7 when *SN* is 60, *MR* is 0.5, and *limit* is 100. From the figure, the ABC algorithm can proceed to the training phase without getting stuck in local minima.

6.4. Effect of control parameters on the ABC algorithm

We performed a post hoc test with ANOVA to explore whether the differences among MR parameters is statistically significant

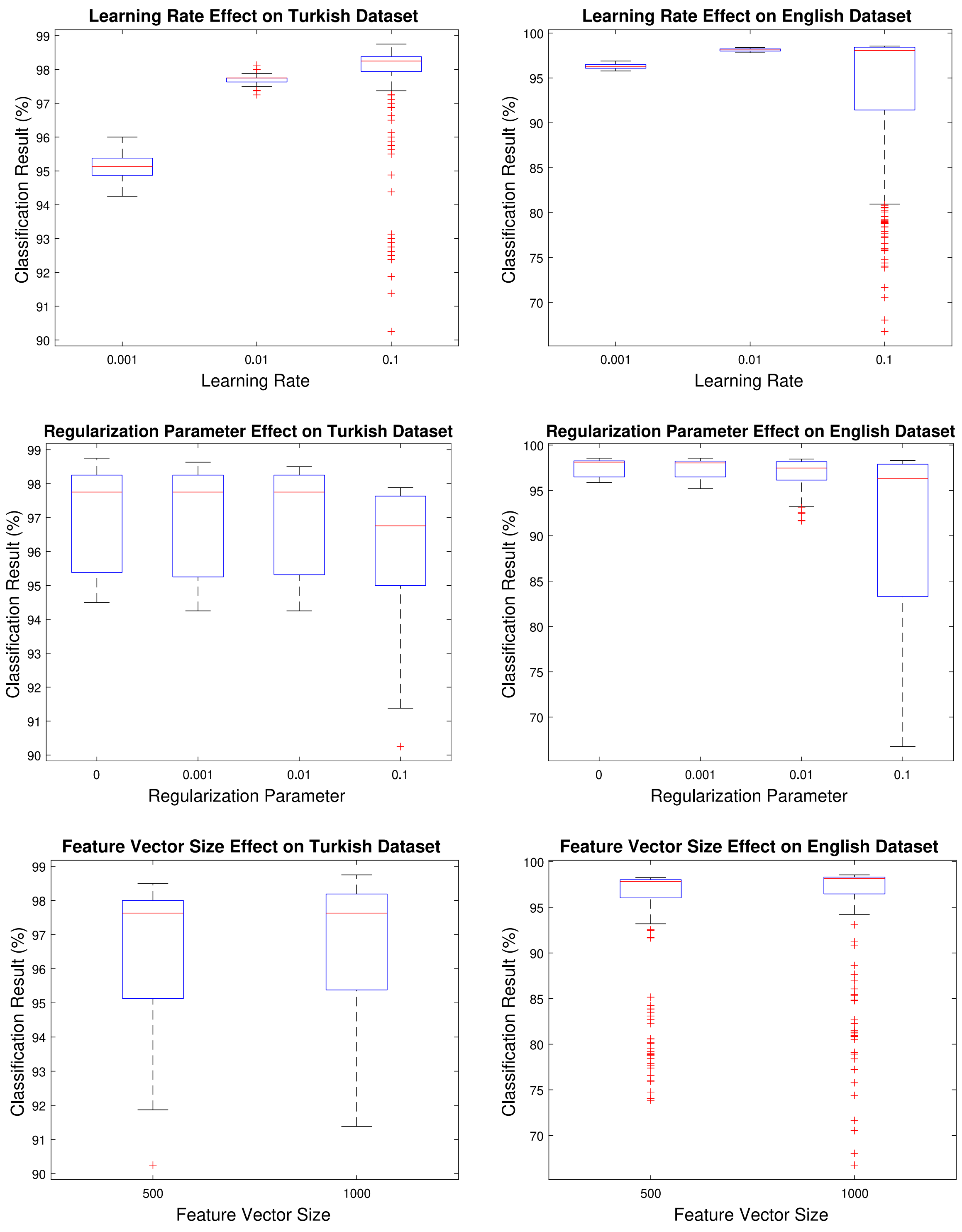

Fig. 6. Effects of parameters and the number of features on the TurkishEmail dataset.

on the results obtained by using the TurkishEmail and CSDMC2010 datasets. Because there are four different groups (MR parameters), we needed a post hoc test to identify which differences between pairs of means were significant. Table 15 shows no statistically significant difference ($Sig = 0.748 > 0.05$) between the group means of the classification accuracies on the Turkish dataset. However, on the English dataset, there is a statistically significant difference ($Sig. = 0.0 < 0.05$) between group means. From Table 17, there are statistically significant differences between the means of 0.05–0.1, 0.05–0.2, and 0.08–0.2 pairs, whereas there is no statistically significant difference between the means of 0.05–0.08 and 0.08–0.1.

Because there are two groups of FVS parameters, we performed an ANOVA test to explore whether the differences between FVS parameters were statistically significant. There are statistically significant differences between the group means of the classification accuracies on both the TurkishEmail and CS-DMC2010 datasets, as seen from Table 16. According to the

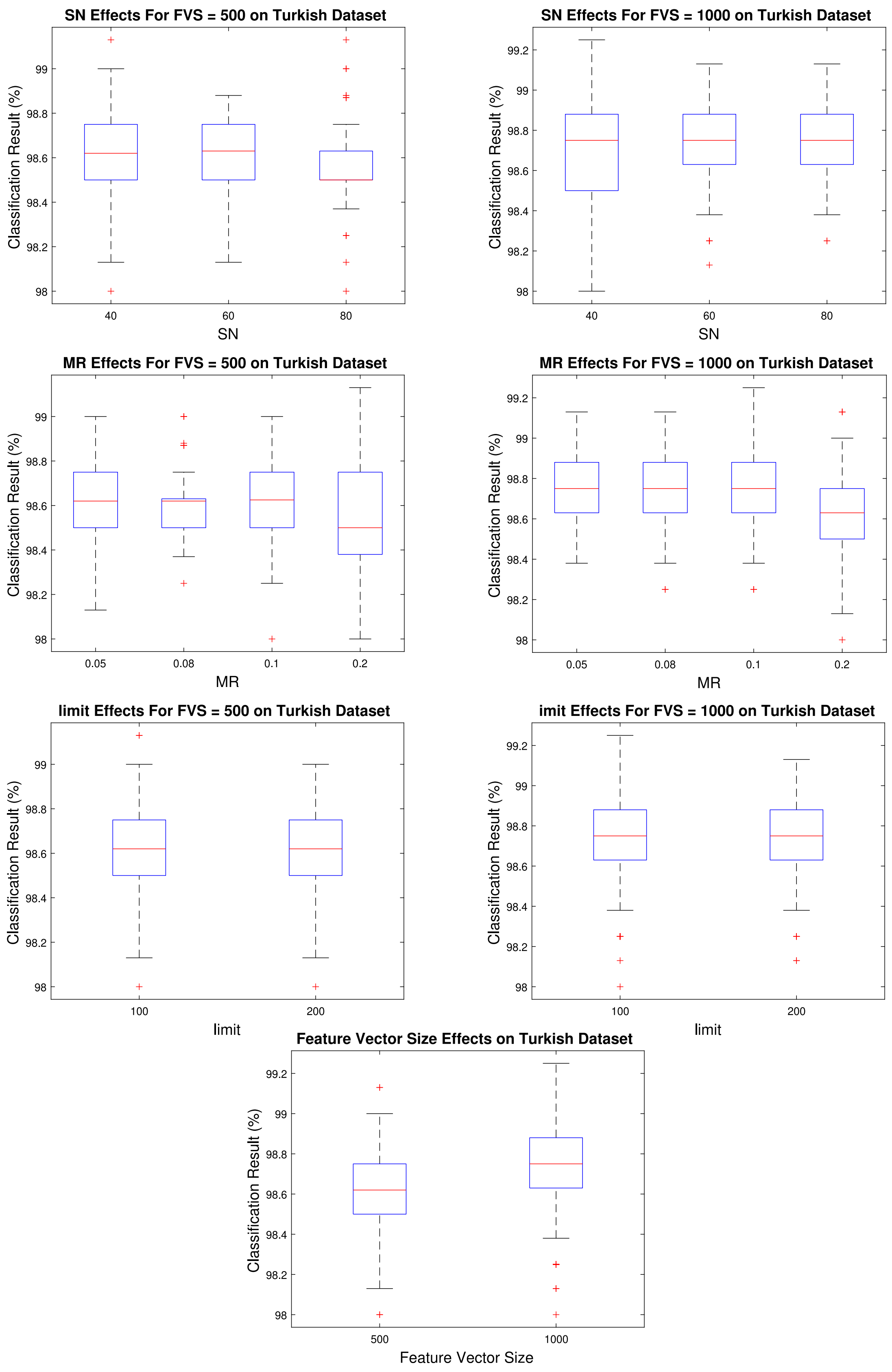

Fig. 7. Convergence graphs of the proposed model in the training phase on the Enron, CSDMC2010, and Turkish Email data sets.

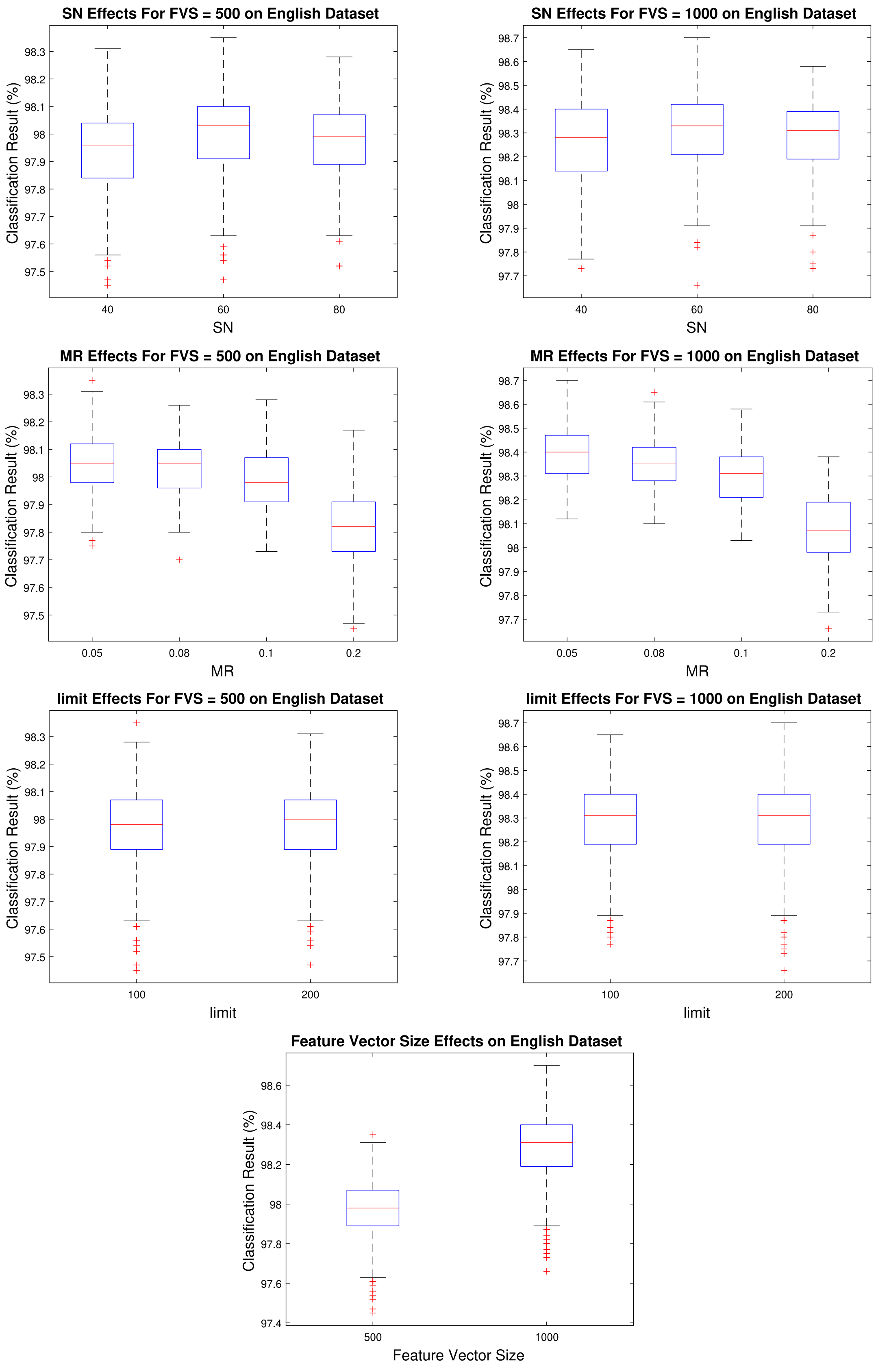

Fig. 8. Effects of parameters and the number of features on the CSDMC2010 dataset.

Table 14
Classification statistics of LR trained by an ABC algorithm on the Enron1 dataset.

SN	*MR*	Feature vector size = 500					Feature vector size = 1000				
		Best	Worst	Median	Mean	Std.	Best	Worst	Median	Mean	Std
40	0.05	98.45%	98.02%	98.21%	98.22%	0.11	98.81%	98.42%	98.49%	98.55%	0.11
	0.1	98.24%	97.87%	98.07%	98.06%	0.10	98.53%	98.05%	98.24%	98.23%	0.10
60	0.05	98.47%	98.05%	98.21%	98.24%	0.11	98.91%	98.49%	98.65%	98.66%	0.10
	0.1	98.30%	97.87%	98.09%	98.09%	0.11	98.58%	98.12%	98.40%	98.40%	0.12
80	0.05	98.30%	97.97%	98.17%	98.15%	0.10	98.75%	98.47%	98.60%	98.61%	0.07
	0.1	98.20%	98.01%	98.12%	98.11%	0.05	98.62%	98.32%	98.47%	98.46%	0.08

ANOVA test results, there is no statistically significant difference either between limit parameters or between the SN and MCN parameters with respect to the classification accuracies on the CSDMC2010 and TurkishEmail datasets.

6.5. *Comparison of classifiers*

Table 18 shows the best and worst classification accuracies obtained from six classifiers on the TurkishEmail and CSDMC2010 datasets for feature vector sizes of 500 and 1000. As seen in Table 18, the LR classifier using the ABC algorithm achieved the best overall results, with success rates of 99.25% and 98.70% on the TurkishEmail and CSDMC2010 datasets, respectively.

Furthermore, we compared the classification performance of ABC-LR to state-of-the-art spam filters tested in a study by Barushka et al. [38]. The classification performances of the spam filters are shown in Table 19. Here, we provide the best accuracy of the spam filters and corresponding FN and FP rates, in addition to the average elapsed time in seconds. We also show the standard deviation values of each result obtained using tenfold cross-validation in Table 19.

The results demonstrate that ABC-LR outperformed the other algorithms on the Enron dataset in terms of classification accuracy and FN rate. ABC-LR also achieved significantly better FN rates than the other methods on the Enron dataset. Comparing FP rates, ABC-LR performed better than all other algorithms on the Enron dataset except for the FDA + NB, random forest, and AdaBoost algorithms. However, these three methods should not be chosen for spam filtering because of their relatively poor detection performance (FN rate).

Finally, the classification accuracy of the proposed model was compared to the results of previous studies using the Enron dataset. In Table 20, we show that ABC-LR achieved better classification accuracy than the other algorithms.

Table 15
ANOVA test result of MR parameters.

		Sum of squares	df	Mean square	F	Sig.
Turkish	Between groups	0.000	3	0.000	0.407	0.748
	Within groups	0.000	116	0.000		
	Total	0.000	119			
CSDMC	Between groups	0.000	3	0.000	42.102	0.000
	Within groups	0.000	116	0.000		
	Total	0.000	119			

Table 16
ANOVA test result of FVS parameters.

		Sum of squares	df	Mean square	F	Sig.
Turkish	Between groups	0.000	1	0.000	4.648	0.035
	Within groups	0.000	58	0.000		
	Total	0.000	59			
CSDMC	Between groups	0.000	1	0.000	172.232	0.000
	Within groups	0.000	58	0.000		
	Total	0.000	59			

7. Conclusion

An LR classifier is an easy and efficient model that is suitable for real-time applications. However, the training algorithm used to obtain its optimal weight and bias values may converge to local minima, and it is sensitive to equal feature weights. In this study, we proposed a novel spam filtering approach that combines the advantages of LR, ABC algorithms, and the tf-idf method. The proposed model was applied to a spam filtering task on three public datasets (TurkishEmail, CSDMC2010 and Enron) in the Turkish and English languages. To account for the language differences, we used two different stemming libraries to obtain word stems and feature vector sizes of 500 and 1000. We investigated two NB algorithms, two SVM algorithms, an LR algorithm trained by gradient descent, and the proposed model on spam filtering and performed numerous experiments. Finally, we compared the classification accuracies of the ABC-LR algorithm against those of the other five algorithms. Experimental results show that the proposed model outperforms the other five algorithms on both datasets and achieves considerable success, with a 99.25% success rate on the TurkishEmail dataset and a 98.70% success rate on the CSDMC2010 dataset. Additionally, our study demonstrated that language complexity can be easily dealt with by considering the morphological structure of the language in the stemming operation.

Furthermore, we used the popular Enron dataset to compare the classification performance of the proposed model (ABC-LR) with that of state-of-the-art methods used in previous studies. We demonstrated that the proposed approach performs better than other methods in terms of classification accuracy and FN rate. The proposed ABC-LR algorithm clearly shows effective performance on imbalanced and nonlinear spam datasets.

The main limitation of the proposed method is that it has high computational complexity compared to other algorithms (except for MLPs, CNNs, and DBB-RDNN-ReL). The computational complexity of ABC-LR can be improved by using parallel computation techniques; further experimentation in this direction is recommended. Further study could use the ABC algorithm to train different classification models, such as SVMs, artificial neural networks, and CNNs. It may also be interesting to compare classification accuracies produced by the ABC algorithm with those by other optimization algorithms, such as PSO, differential evolution, AIS, and ALO, on the spam filtering problem. The effects of normalization and feature selection methods could also be investigated in a future study.

Table 17
Post hoc test result of MR parameters.

Dependent var.	(I)MR	(J)MR	Mean diff. (I–J)	Std. error	Sig.	Lower bound	Upper bound
CSDMC Dataset Tukey HSD	0.05	0.08	0.0002867	0.0002848	0.746	−0.000456	0.001029
		0.10	0.0008733*	0.0002848	0.014	0.000131	0.001616
		0.20	0.0028967*	0.0002848	0.000	0.002154	0.003639
	0.08	0.05	−0.0002867	0.0002848	0.746	−0.001029	0.000456
		0.10	0.0005867	0.0002848	0.172	−0.000156	0.001329
		0.20	0.0026100*	0.0002848	0.000	0.001868	0.003352
	0.10	0.05	−0.0008733*	0.0002848	0.014	−0.001616	−0.000131
		0.08	−0.0005867	0.0002848	0.172	−0.001329	0.000156
		0.20	0.0020233*	0.0002848	0.000	0.001281	0.002766
	0.20	0.05	−0.0028967*	0.0002848	0.000	−0.003639	−0.002154
		0.08	−0.0026100*	0.0002848	0.000	−0.003352	−0.001868
		0.10	−0.0020233*	0.0002848	0.000	−0.002766	−0.001281

*Denotes that there is a statistically significant difference.

Table 18
Comparison of the best and worst classification accuracies on for the TurkishEmail and CSDMC2010 datasets.

Classification method	TurkishEmail dataset				CSDMC2010 dataset			
	FVS = 500		FVS = 1000		FVS = 500		FVS = 1000	
	Best	Worst	Best	Worst	Best	Worst	Best	Worst
Gaussian NB	95.38%	95.38%	96.63%	96.63%	93.71%	93.71%	95.92%	95.92%
Multinomial NB	96.88%	95.63%	97.62%	97.12%	90.93%	88.89%	94.55%	92.02%
Linear SVM	98.63%	93.00%	98.88%	93.75%	98.10%	92.30%	98.42%	94.08%
SVM with RBF kernel	98.62%	53.75%	98.75%	54.00%	98.58%	68.21%	98.63%	68.21%
LR by gradient descent	98.50%	90.25%	98.75%	91.38%	98.26%	73.85%	98.56%	66.75%
The proposed model	99.13%	98.00%	99.25%	98.13%	98.35%	97.45%	98.70%	97.66%

Table 19
Comparison of ABC-LR's accuracy, FN, FP, and average elapsed training time in seconds with those of algorithms described in a previous study [38] on the Enron 1 dataset.

Method	Accuracy ± Std	FN ± Std	FP ± Std	Avg time ± Std
MDL [38,46]	95.67 ± 1.13	0.0107 ± 0.0105	0.0566 ± 0.0165	10.1864 ± 1.4689
FDA + NB [38,47]	91.29 ± 1.18	0.1209 ± 0.0163	0.0045 ± 0.0057	0.8592 ± 0.0374
FDA + SVM [38,47]	96.66 ± 0.84	0.0459 ± 0.0178	0.0283 ± 0.0092	5.4755 ± 1.1081
IL-C4.5 [38,48]	93.35 ± 1.29	0.0608 ± 0.0207	0.0688 ± 0.0159	40.0021 ± 3.6369
Voting [38,49]	97.20 ± 1.06	0.0247 ± 0.0118	0.0294 ± 0.0130	34.3022 ± 2.0173
Random forest [38,50]	98.05 ± 0.57	0.0178 ± 0.0121	0.0201 ± 0.0073	28.0200 ± 0.3813
AdaBoost [38]	78.76 ± 1.15	0.6838 ± 0.0340	0.0200 ± 0.0800	2.7871 ± 0.0530
LR [38]	94.54 ± 1.04	0.0668 ± 0.0200	0.0496 ± 0.0125	4.7599 ± 0.8967
AIRS2Parallel [38]	71.36 ± 7.65	0.1220 ± 0.1679	0.3535 ± 0.1519	21.2569 ± 0.5605
MLP [38]	96.29 ± 2.84	0.0501 ± 0.0823	0.0318 ± 0.0361	347.9439 ± 9.1491
kNN [38]	91.36 ± 1.34	0.0378 ± 0.0158	0.1062 ± 0.0177	0.0018 ± 0.0039
CNN [38]	97.47 ± 0.87	0.0347 ± 0.0193	0.0215 ± 0.0079	170.7381 ± 28.4784
DBB-RDNN-Rel [38]	98.76 ± 0.57	0.0017 ± 0.0018	0.0212 ± 0.0101	183.5865 ± 28.3987
ABC-LR	98.91 ± 0.87	0.0005 ± 0.0011	0.0210 ± 0.0176	113.5650 ± 1.2632

Table 20
Comparison of ABC-LR's accuracy with those of previous methods on the Enron 1 dataset.

Study	Method	Accuracy
[51]	Deep belief networks	97.43%
[52]	AIS	90.00%
[18]	Multivariate Bernoulli NB	94.79%
[53]	Distinguishing feature selector	94.35%
[54]	Minimum description length	95.56%
[55]	Bagged RF	97.75%
[56]	Enhanced genetic programming	94.10%
[57]	RF	96.39%
[58]	Relief + NB	96.30%
[59]	k means + SVM	97.35%
[60]	Natural language toolkit NB	94.70%
[25]	Incremental SVM	96.86%
[61]	Boosted NB + SVM	95.60%
[38]	DBB-RDNN-ReL	98.76%
This study	ABC-LR	98.91%

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Bilge Kagan Dedeturk: Conceptualization, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing. **Bahriye Akay:** Conceptualization, Investigation, Methodology, Resources, Project administration, Supervision, Writing.

Acknowledgment

This research was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under project number 116E947.

References

[1] G. V. Cormack, Email spam filtering: A systematic review, Found. Trends Inf. Retr. 1 (2008) 335–455, http://dx.doi.org/10.1561/1500000006.

[2] Email statistics report, 2019-2023, in: Executive summary, The Radicati Group, 2019.

[3] Global e-mail spam rate from 2012 to 2018, Tech. rep., Statista, 2018.

[4] Internet Security Threat Report, Tech. rep., Symantec, 2018.

[5] Spam and phishing in q3 2018, Spam and phishing reports, Kaspersky, 2018.

[6] A. Bhowmick, S.M. Hazarika, E-mail spam filtering: A review of techniques and trends, in: Lecture Notes in Electrical Engineering, Springer Singapore, 2017, pp. 583–590.

[7] L. Ozgur, T. Gungor, F. Gurgen, Adaptive anti-spam filtering for agglutinative languages: a special case for turkish, Pattern Recognit. Lett. 25 (16) (2004) 1819–1831.

[8] Y. Han, M. Yang, H. Qi, X. He, S. Li, The improved logistic regression models for spam filtering, in: 2009 International Conference on Asian Language Processing, 2009, pp. 314–317, http://dx.doi.org/10.1109/IALP.2009.74.

[9] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report - TR06, Technical Report, Erciyes University, 2005.

[10] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Appl. Math. Comput. 214 (2009) 108–132.

[11] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Inform. Sci. 192 (2012) 120–142.

[12] T. Gungor, A. Ciltik, Developing methods and heuristics with low time complexities for filtering spam messages, in: Natural Language Processing and Information Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 35–47.

[13] D. Heckerman, E. Horvitz, M. Sahami, S. Dumais, A Bayesian approach to filtering junk e-mail, in: AAAI Workshop on Learning for Text Categorization, 1998, pp. 55–62.

[14] I. Androutsopoulos, J. Koutsias, K. Chandrinos, G. Paliouras, C. Spyropoulos, An evaluation of naive Bayesian anti-spam filtering, 2000, CoRR cs.CL/0006013.

[15] V. Metsis, I. Androutsopoulos, G. Paliouras, Spam filtering with naive bayes - which naive bayes? in: Third Conference on Email and Anti-Spam (ceas, 2006.

[16] I. Androutsopoulos, J. Koutsias, K.V. Chandrinos, C.D. Spyropoulos, An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages, in: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, in: SIGIR '00, ACM, New York, NY, USA, 2000, pp. 160–167, http://dx.doi.org/10.1145/345508.345569.

[17] N.F. Rusland, N. Wahid, S. Kasim, H. Hafit, Analysis of naive bayes algorithm for email spam filtering across multiple datasets, in: IOP Conference Series: Materials Science and Engineering, Vol. 226, IOP Publishing, 2017, p. 012091.

[18] T.A. Almeida, J. Almeida, A. Yamakami, Spam filtering: how the dimensionality reduction affects the accuracy of naive bayes classifiers, J. Internet Serv. Appl. 1 (3) (2011) 183–200.

[19] W. Feng, J. Sun, L. Zhang, C. Cao, Q. Yang, A support vector machine based naive bayes algorithm for spam filtering, in: 2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC), 2016, pp. 1–8.

[20] V. N. Vapnik, The nature of statistical learning theory, Nat. Stat. Learn. Theory 6 (1995) http://dx.doi.org/10.1007/978-1-4757-2440-0.

[21] H. Drucker, D. Wu, V. Vapnik, Support vector machines for spam categorization, IEEE Trans. Neural Netw. 10 5 (1999) 1048–1054.

[22] O. Amayri, N. Bouguila, A study of spam filtering using support vector machines, Artif. Intell. Rev. 34 (1) (2010) 73–108, http://dx.doi.org/10.1007/s10462-010-9166-x.

[23] D. Sculley, G.M. Wachman, Relaxed online svms for spam filtering, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, in: SIGIR '07, ACM, New York, NY, USA, 2007, pp. 415–422, http://dx.doi.org/10.1145/1277741.1277813.

[24] B. Yu, Z. ben Xu, A comparative study for content-based dynamic spam classification using four machine learning algorithms, Knowl.-Based Syst. 21 (4) (2008) 355–362, http://dx.doi.org/10.1016/j.knosys.2008.01.001.

[25] G. Sanghani, K. Kotecha, Personalized spam filtering using incremental training of support vector machine, in: 2016 International Conference on Computing, Analytics and Security Trends (CAST), 2016, pp. 323–328, http://dx.doi.org/10.1109/CAST.2016.7914988.

[26] J. Goodman, S.W.-t. Yih, Online discriminative spam filter training, in: Proceedings of the 3rd Conference on Email and Anti-Spam, Proceedings of the 3rd Conference on Email and Anti-Spam, CEAS, 2006.

[27] M.-w. Chang, W.-t. Yih, C. Meek, Partitioned logistic regression for spam filtering, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '08, ACM, New York, NY, USA, 2008, pp. 97–105, http://dx.doi.org/10.1145/1401890.1401907.

[28] I. Idris, A. Selamat, Improved email spam detection model with negative selection algorithm and particle swarm optimization, Appl. Soft Comput. 22 (2014) 11–27, http://dx.doi.org/10.1016/j.asoc.2014.05.002.

[29] R. Chikh, S. Chikhi, Clustered negative selection algorithm and fruit fly optimization for email spam detection, J. Ambient Intell. Hum. Comput. 10 (1) (2019) 143–152, http://dx.doi.org/10.1007/s12652-017-0621-2.

[30] A.J. Saleh, A. Karim, B. Shanmugam, S. Azam, K. Kannoorpatti, M. Jonkman, F.D. Boer, An intelligent spam detection model based on artificial immune system, Information 10 (2019) 209.

[31] A.A. Naem, N.I. Ghali, A.A. Saleh, Antlion optimization and boosting classifier for spam email detection, Future Comput. Inf. J. 3 (2) (2018) 436–442, http://dx.doi.org/10.1016/j.fcij.2018.11.006.

[32] T. Fawcett, "in vivo" spam filtering: A challenge problem for kdd, SIGKDD Explor. Newsl. 5 (2) (2003) 140–148, http://dx.doi.org/10.1145/980972.980990.

[33] S. Tutun, S. Khanmohammadi, L. He, C.-A. Chou, A meta-heuristic lasso model for diabetic readmission prediction, in: Conference: Industrial and Systems Engineering Research Conference (ISERC), 2016.

[34] S. Ergin, E.S. Gunal, H. Yigit, R.S.T. Aydin, Turkish anti-spam filtering using binary and probabilistic models, in: 2nd World Conference on Information Technology, WCIT-2011, 2012, pp. 1007–1012.

[35] Tin Kam Ho, M. Basu, Complexity measures of supervised classification problems, IEEE Trans. Pattern Anal. Mach. Intell. 24 (3) (2002) 289–300, http://dx.doi.org/10.1109/34.990132.

[36] S. Bird, E. Klein, E. Loper, Natural Language Processing with Python, first ed., O'Reilly Media, Inc., 2009.

[37] T. Golub, Modernized mathematical model of text document classification, in: CMIS, 2019, pp. 607–617.

[38] A. Barushka, P. Hajek, Spam filtering using integrated distribution-based balancing approach and regularized deep neural networks, Appl. Intell. 48 (10) (2018) 3538–3556, http://dx.doi.org/10.1007/s10489-018-1161-y.

[39] L.H. Patil, M. Atique, A novel approach for feature selection method tf-idf in document clustering, in: 2013 3rd IEEE International Advance Computing Conference (IACC), 2013, pp. 858–862, http://dx.doi.org/10.1109/IAdCC.2013.6514339.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[41] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, G. Varoquaux, API design for machine learning software: experiences from the scikit-learn project, in: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, 2013, pp. 108–122.

[42] C.-W. Hsu, C.-C. Chang, C.-J. Lin, A practical guide to support vector classification, Tech. rep., Department of Computer Science, National Taiwan University, 2003, URL http://www.csie.ntu.edu.tw/~cjlin/papers.html.

[43] C.-C. Chang, C.-J. Lin, Libsvm: A library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 27:1–27:27, http://dx.doi.org/10.1145/1961189.1961199.

[44] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Inf. Sci. - ISCI 192 (2010).

[45] B. Akay, D. Karaboga, Parameter tuning for the artificial bee colony algorithm, Lect. Notes Artif. Intell. 5796 (2009) 608–619.

[46] T.A. Almeida, A. Yamakami, Compression-based spam filter, Secur. Commun. Netw. 9 (2016) 327–335.

[47] M.V. Aragão, E.P. Frigieri, C.A. Ynoguti, A.P. Paiva, Factorial design analysis applied to the performance of sms anti-spam filtering systems, Expert Syst. Appl. 64 (2016) 589–604, http://dx.doi.org/10.1016/j.eswa.2016.08.038.

[48] J.-J. Sheu, K.-T. Chu, N.-F. Li, C.-C. Lee, An efficient incremental learning mechanism for tracking concept drift in spam filtering, PLOS ONE 12 (2) (2017) 1–17, http://dx.doi.org/10.1371/journal.pone.0171518.

[49] R. Abooraig, S. Nawasrah, H. Najadat, N. Abdulla, Spam detection for mobile short messaging service using data mining classifiers, Int. J. Comput. Sci. Inf. Secur. (IJCSIS) (2016) 511–517.

[50] Z. Khorshidpour, S. Hashemi, A. Hamzeh, Evaluation of random forest classifier in security domain, Appl. Intell. 47 (2017) 558–569.

[51] G. Tzortzis, A. Likas, Deep belief networks for spam filtering, in: 19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007), Vol. 2, 2007, pp. 306–309, http://dx.doi.org/10.1109/ICTAI.2007.65.

[52] A. Abi-Haidar, L. Rocha, Adaptive spam detection inspired by the immune system, 2008.

[53] A.K. Uysal, S. Gunal, A novel probabilistic feature selection method for text classification, Knowl.-Based Syst. 36 (2012) 226–235, http://dx.doi.org/10.1016/j.knosys.2012.06.005.

[54] T.A. Almeida, A. Yamakami, Occam's razor-based spam filter, J. Internet Serv. Appl. 3 (2012) 245–253.

[55] R. Shams, R.E. Mercer, Personalized spam filtering with natural language attributes, in: 2013 12th International Conference on Machine Learning and Applications, Vol. 2, 2013, pp. 127–132, http://dx.doi.org/10.1109/ICMLA.2013.117.

[56] S.K. Trivedi, S. Dey, An enhanced genetic programming approach for detecting unsolicited emails, in: 2013 IEEE 16th International Conference on Computational Science and Engineering, 2013, pp. 1153–1160, http://dx.doi.org/10.1109/CSE.2013.171.

[57] R. Mishra, R. Thakur, Analysis of random forest and naive bayes for spam mail using feature selection catagorization, Int. J. Comput. Appl. 80 (2013) 42–47, http://dx.doi.org/10.5120/13844-1670.

[58] S.K. Trivedi, S. Dey, A comparative study of various supervised feature selection methods for spam classification, in: Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies, in: ICTCS '16, Association for Computing Machinery, New York, NY, USA, 2016, http://dx.doi.org/10.1145/2905055.2905122.

[59] D. Hassan, Investigating the effect of combining text clustering with classification on improving spam email detection, in: A.M. Madureira, A. Abraham, D. Gamboa, P. Novais (Eds.), Intelligent Systems Design and Applications, Springer International Publishing, Cham, 2017, pp. 99–107.

[60] K. Chhogyal, A. Nayak, An empirical study of a simple naive bayes classifier based on ranking functions, in: B.H. Kang, Q. Bai (Eds.), AI 2016: Advances in Artificial Intelligence, Springer International Publishing, Cham, 2016, pp. 324–331.

[61] S.K. Trivedi, S. Dey, A combining classifiers approach for detecting email spams, in: 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2016, pp. 355–360, http://dx.doi.org/10.1109/WAINA.2016.127.