IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON NEW TRENDS IN BRAIN SIGNAL PROCESSING AND ANALYSIS

Received January 8, 2019, accepted January 14, 2019, date of current version February 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2894871

A Particle Swarm Optimized Learning Model of Fault Classification in Web-Apps

DEEPAK KUMAR JAIN[1], AKSHI KUMAR[2], SAURABH RAJ SANGWAN[2], GIA NHU NGUYEN[3], AND PRAYAG TIWARI[4]

[1]Key Laboratory of Intelligent Air-Ground Cooperative Control for Universities in Chongqing, College of Automation, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[2]Department of Computer Science and Engineering, Delhi Technological University, New Delhi 110042, India
[3]Graduate School, Duy Tan University, Danang 550000, Vietnam
[4]Department of Information Engineering, University of Padova, Padua, Italy

Corresponding author: Gia Nhu Nguyen (nguyengianhu@duytan.edu.vn)

This work was supported in part by the Key Laboratory of Intelligent Air-Ground Cooperative Control for Universities in Chongqing and the Key Laboratory of Industrial IoT and Networked Control, Ministry of Education, College of Automation, Chongqing University of Posts and Telecommunications, Chongqing, China.

ABSTRACT The term web-app defines the current dynamic pragmatics of the website, where the user has control. Finding faults in such dynamic content is challenging, as to whether the fault is exposed or not depends on its execution path. Moreover, the complexity and uniqueness of each web application make fault assessment an extremely laborious and expensive task. Also, artificial fault injection models are run in controlled and simulated environments, which may not be representative of the real-world fault data. Classifying faults can intelligently enhance the quality of the web-apps by the assessment of probable faults. In this paper, an empirical study is conducted to classify faults in bug reports of three open-source web-apps (*qaManager*, *bitWeaver*, and *WebCalendar*) and reviews of two play store web-apps (*Dineout: Reserve a Table and Wynk Music*). Five supervised learning algorithms (naïve Bayesian, decision tree, support vector machines, K-nearest neighbor, and multi-layer perceptron) have been first evaluated based on the conventional term frequency–inverse document frequency (tf-idf) feature extraction method, and subsequently, a feature selection method to improve classifier performance is proposed using particle swarm optimization (a nature-inspired, meta-heuristic algorithm). This paper is a preliminary exploratory study to build an automated tool, which can optimally categorize faults. The empirical analysis validates that the particle swarm optimization for feature selection in fault classification task outperforms the tf-idf filter-based classifiers with an average accuracy gain of about 11% and nearly 26% average feature reduction. The highest accuracy of 93.35% is shown by the decision tree after feature selection.

INDEX TERMS Classification, fault, feature selection, particle swarm, web-apps.

I. INTRODUCTION

With the increasing size of indexed Web, superior technology, and optimal browser performance, the development of Web has seen significant transformation from being an anachronistic static content repository to a turbulent, interactive, responsive content space. The websites now rely on programmatic user input and data processing. The term web-based applications or simply web-app [1] defines the current dynamic pragmatics of the website where the user has control. Technologically, the current generation websites are more like web-based software which store data/interact with a database on the back end, and process business logic and information in a more convoluted way. They have a web interface but web development here is just not limited to developing an alluring interface but creating web-based software. Thus, the web-based software development primarily consists of three ingredients, namely the development of websites, web application development and development of web services [2].

"Agile development practices", "big-data", "security", "open source" and "customer-first design" are some of the key terms that characterize the latest technology trends in the software development. A typical web application development workflow is similar to the conventional software development (Figure 1). It involves five phases, (i) *brainstorm:* for requirement analysis (ii) *design:* design document & prototype (iii) *development:* iterations, demo and feedback

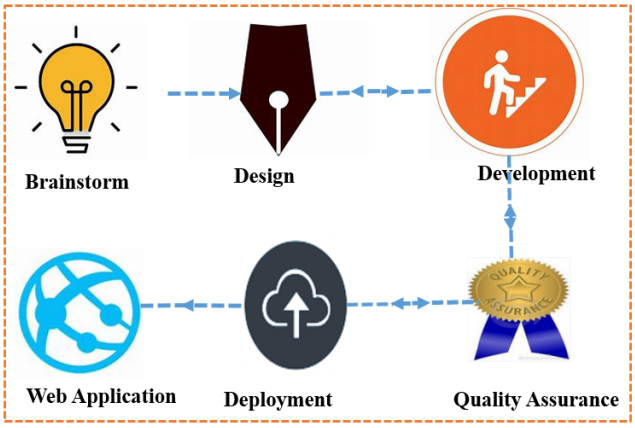

FIGURE 1. Web-app development workflow.

(iv) *quality assurance*: identify defects and resolve bugs (v) *deployment:* production and technical support.

Web quality is defined as the degree to which the web-based software meets the specified requirements, is accessible, provides the reliable information and meets the user needs & expectations [2]. Quality assurance (QA) popularly known as QA testing is a systematic process of determining whether a product or service meets specified requirements and customer expectations. This process-driven approach is the key to ensure the performance and reliability of the product. QA's primary goal is tracking and resolving deficiencies prior to product release, that too in a proactive way. That is, the purpose of QA is to prevent defects from entering into the system. Thus, the quality assessment models in this new development setting call for a set of acceptance criteria which define accessibility and usability of the web-apps, demonstrating its effectiveness in terms of user-experience. The acceptance of websites/apps by the end-user depends on a variety of factors. The most measurable aspect of software quality is the number of faults, or bugs, that are discovered in a software product. It is an absolute quantifier of quality. Formally, a fault is defined as an ''incorrect step, process, or data definition in a computer program'' [3]. For example, attributes such as dead-links or browser compatibility [2] are direct indicators of faults and a quality compromise.

The source code and bug reports for the open-source projects are readily available but are awfully structured and unlabeled. Similarly, the Google and Apple play store allow users to give feedback in the form of reviews. An app review typically includes star rating followed by a comment. Negative opinion polarity within these reviews is strong indicator of a fault/drawback/short-coming. For example, the comment *"The link for few songs are dead…play now and download both not working for them!!! The text is not readable as the font is too small. Moreover, the premium version is a waste of money as compared to…."*, clearly conveys the opinion of the reviewer(negative in this case) and the aspects of negative opinion. These reviews possess a vital source of information that can be used by the app developers and the vendors for de-bugging and version control. But here too, most of the reviews are in an unstructured form and mining useful analytical information from them requires a great effort. Pro-actively finding categories of faults in such open-source projects and real-time data that too manually is arduous and expensive. Researchers have recognized the limits of manual fault classification and have investigated automation solutions. Empirical fault detection is one such promising research direction which is based on fault classification as an imperative prerequisite task [4]. An automated tool to classify faults into pre-defined categories can assist in fault-based testing of web-apps. The primary purpose of any such tool is to prevent faults and find as many faults as possible, as early as possible.

The vital sub-task of the fault classification process is feature extraction, which converts the input data (unstructured textual data indicative of faults), into an array of representative features. Commonly, the feature extraction task is done using intrinsic 'filtering' methods which are fast, classifier-independent methods that rank features according to predetermined numerical functions based on the measure of the ''importance'' of the terms. A variety of scoring functions such as, tf-idf, chi-square, mutual information, information gain, cross-entropy etc., have been used as statistical measures to pick features with the highest scores [5]. Further, past literature conforms that an optimal feature selection [6] improves the classifier performance (in terms of speed, predictive power and simplicity of the model), reduces dimensionality, removes noise and helps visualizing the data for model selection. In feature selection the features are kept intact and n best features are chosen among them, removing the redundant and co-linear features. This sub-task of selecting the relevant subset of features and discarding the non-essential ones is computationally challenging and expensive task. Population-based meta-heuristics, especially the ones inspired by nature have been proposed for feature selection in relevant and prominent literature work as wrapper methods to select the best possible subset of features for a given model.

Swarm intelligence algorithms (SI) are contemporary computational and behavioral metaphors for solving search and optimization problems which take collective biological patterns provided by social insects (ants, termites, bees, wasps, moths etc.) and other animal societies (fish, birds, grey wolves etc.) as stimulus to model algorithmic solutions. Several algorithms inspired by natural phenomena have been proposed in the past years and among them, some meta-heuristic search algorithms with population-based framework have shown satisfactory capabilities to handle high dimension optimization problems. The work presented in this paper is an insight to this research trend which comprehends the adaptive learning and collective intelligence behavioral models of swarm-based algorithms. Swarm-based feature optimization using particle swarm optimization (PSO) is demonstrated to

cater to the dimensionality; complexity and fuzziness in the unstructured data.

The research presented in this paper, is an empirical study to put forward an optimized learning model to reduce the feature dimensionality in order to optimize fault prediction in real-time web-apps. Bug reports of three open-source web-app projects (*qaManager*[1], *bitWeaver*,[2] *WebCalendar*)[3] and reviews from two play store applications (*Dineout: Reserve a Table*,[4] *Wynk Music*)[5] have been considered to intelligently mine faults which are annotated based on seven categories as given by Sampath *et al*. [7]. A comparative analysis using five supervised learning algorithms, namely naïve Bayesian (NB), decision tree (DT), support vector machine (SVM), K-nearest neighbor (K-NN) and multi-layer perceptron (MLP) is done to find the best predictive classifier. Thus the contribution of this research is to build an optimal fault prediction model as follows:

- Implement five supervised learning algorithms to predict faults using tf-idf feature extraction: NB, DT, SVM, K-NN, MLP
- Implement five supervised learning algorithms using feature selection method: tf-idf +PSO
- Performance analysis on the basis of accuracy

The predictive model will give insights to the testing practitioners to seed similar types of faults for comprehending fault-based testing of Web-Apps. This study does not compare various feature selection algorithms, but it demonstrates the benefit of adding the feature selection optimization process together with the fault classification task to enhance the accuracy of classifier.

The organization of the paper is as follows: Section 2 discusses the related work in the domain of research. Section 3 describes the system architecture along with the details of the proposed optimal predictive learning model using particle swarm optimization. Section 4 discusses the results and finally, section 5 presents the conclusion of the empirical study.

II. RELATED WORK

Few pertinent studies have been conducted in the area of fault classification in web applications. Sampath *et al.* [7] conducted experiments for fault detection and by seeding realistic faults and classified them into five categories namely data store faults, logic faults, form faults, appearance faults and link faults. Guo and Sampath [4] have carried out an exploratory study on web application fault classification using the induction method wherein they state the classification dimension and used the work in [7] as the baseline. They used the actual location of the fault as a dimension for classification taking into consideration presentation, logic and data store faults. They defined a new fault type known as compatibility fault and fine grained the classification of logic faults into browser interaction faults, session faults, paging faults, server side faults, encoding-decoding faults, locale faults and others and calculated the frequency of faults on Roller Weblogger and qaManager.

Elbaum *et al.* [8] seeded the three types of faults namely scripting, forms and database query faults to evaluate the performance of web testing techniques. Li and Tian [9] adapted an orthogonal defect classification approach (ODC) like problems related to timing, interface or algorithms. Kumar *et al.* [10] have used supervised machine learning techniques on three open source web applications. They used area under ROC curve to analyze and compare the performance of various machine learning techniques and reached the conclusion that multinomial naïve Bayesian gave the best results. A review on the application of computational evolutionary method, the genetic algorithm to automatically search software errors was given by Mantere and Alander [11]

III. SYSTEM ARCHITECTURE

The longer the fault goes without detection, the more expensive the fault is to repair. Fault classification intends to offer feedback about the web development process. Based on this, this research puts forward a fault prediction model using optimal feature selection. A variety of supervised learning algorithms are evaluated to find the best model for fault classification. The goal of using classification schemes is to devise a preventive strategy for finding as many faults as possible and that too as early as possible.

The preliminary step is to gather the required data, which are: bug-reports of open source web-apps and reviews of play store apps. Pre-processing is then done for cleaning the dataset from noise. Noise here basically connotes the language irregularities often present in text, as this noisy and unstructured data affects the quality of the fault classification task. Thus, after pre-processing, the representative features are extracted using the tf-idf filter. Particle swarm optimization (PSO) is then applied on this resulting matrix to generate an optimal feature matrix. These optimal features are then used to train the classifier. The figure 2 shows the systematic flow of the model.

The following sub-sections expound the details:

A. DATASET ACQUISITION

To evaluate the system two types of datasets have been considered. Firstly, three open source web-app projects, namely *qaManage*r, *bitWeaver* and *WebCalendar*have been considered. The bug reports for the same have been acquired from sourceforge.net. A bug is an evidence of fault in the program. And so considering the bug report is an obvious choice for fault diagnosis. We deliberately consider Java/PHP based applications as both these languages are the most popular and widely used ones for web development. The applications with larger number of closed bugs were considered since

[1]qaManager: https://sourceforge.net/projects/qamanager/
[2]bitWeaver: https://sourceforge.net/projects/bitweaver/
[3]WebCalendar: https://sourceforge.net/projects/webcalendar/
[4]Dineout App: https:// play.google.com/ store/ apps/ details?id=com.dineout.book&hl=en_IN
[5]Wynk Music App: https://itunes.apple.com/lk/app/wynk-music/id845083955?mt=8

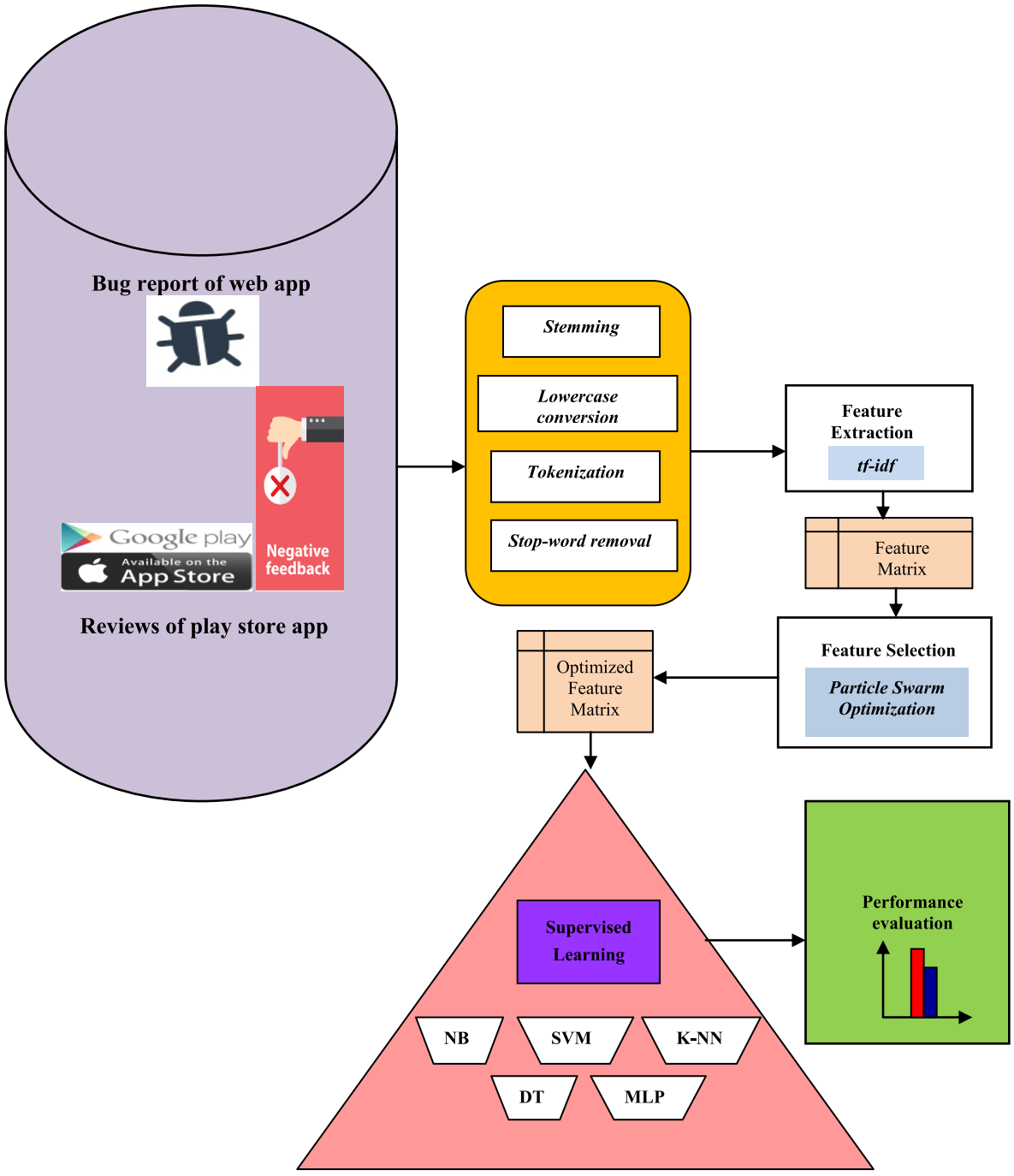

FIGURE 2. System Architecture.

comments given in the closed bug section facilitated easy bug identification. The number of total bugs was also a significant selection criterion.

Secondly, reviews of two play store (1 Google and 1 Apple) apps, namely, *Dineout: Reserve a Table*and *Wynk Music* were acquired. The *Dineout* app had 1135 reviews whereas the *Wynk Music* app has 8470 reviews. The reviews typically include star rating followed by a comment. Negative opinion polarity within these reviews is strong indicator of a fault/drawback/short-coming. These reviews possess a vital source of information that can be used by the app developers and the vendors for de-bugging and version control. Thus, these reviews are analyzed for negative opinion polarity using AFINN-111 sentiment lexicon, which is a list of 2477 English words labeled with sentiment strength [12]. Sentiment refers to the use of polarities (positive and/or negative) in written text [13]–[16]. Each word is assigned with an integer in a range of polarity from -5 up to $+5$, negative to positive.

It includes a number of words frequently used on the Internet such as LOL (Laughing Out Loud), which are indicative of emotions of the user, especially on social media portals. The negative reviews were then analyzed for faults.

Five different web-apps were especially considered to determine the robustness of the learning model. To analyze the faults within the application's bug report and negative reviews, the bugs were scraped using Google Web Scraper[6] and the reports was annotated for faults from seven categories as given by Sampath *et al.* [7]. The categorization was based on physical location of fault as data store, form, logic, link, appearance, compatibility and others. (Figure 3).

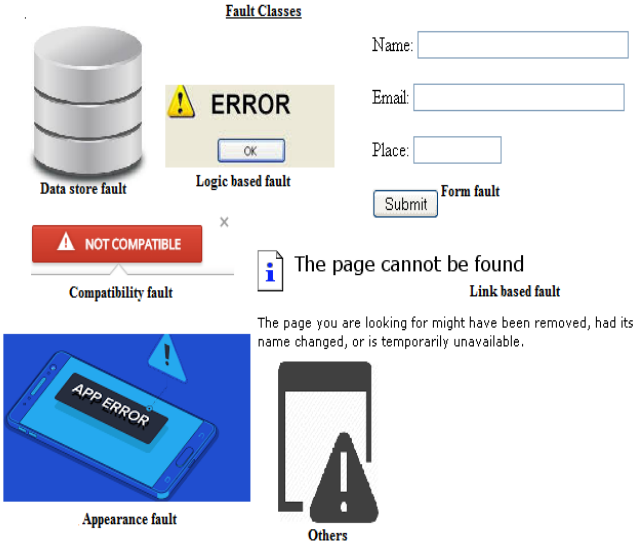

FIGURE 3. Fault categories.

The following table 1 depicts the number of faults for the five web-apps.

TABLE 1. No. of faults in Web-Apps.

Web Apps / Faults	qaMa nager	Bitwea ver	Webcale ndar	Dineou t	Wynk Music
Logic	166	217	1199	34	102
Appearance	29	83	375	15	65
Data Store	19	81	170	22	598
Link	12	51	79	18	234
Form	20	5	5	5	8
Compatibility	5	20	72	24	74
Others	39	75	268	34	354
Total	290	532	2168	152	1435

B. PRE-PROCESSING

To extract the structured text for analytics from the unstructured bug reports/reviews, pre-processing is done [17]. The bag-of-words model [18] is used to transform and represent the text. The method takes into account the words and their frequency of occurrence in the sentence or the document. The data is firstly tokenized to identify tokens and then cleaned by removing punctuations, numbers, special characters, stop words. Stemming is carried out too, to reduce words to their root word. University of Glasgow stop word list[7]and the Porter stemming algorithm [19] are used for stop-word removal and stemming respectively.

C. FEATURE EXTRACTION AND SELECTION

In this work, tf-idf is used a filter method whereas, the PSO is used for optimal feature subset selection. These are briefly described as follows:

The conventional term frequency - inverse document frequency (tf-idf) method is used for calculating the weights and extracting the features. "Term frequency - inverse document frequency is a conventional statistical weight which measures how important a word is to a document" [20]. Moreover, it checks how relevant the keyword is throughout the corpus [21], [22].

The term frequency, tf (t,d) is the raw count of a term in a document and is calculated using equation (1)

$$\text{tf}\,(\text{t, d}) = \frac{\text{No. of times term t appears in a document d}}{\text{Total no. of terms in the document}} \quad (1)$$

The inverse document frequency, idf (t, D) is the measure of how much information is provided by a specific word or term, i.e. whether the word is rare or common across the corpus. idf is calculated using equation (2)

$$\text{idf}\,(\text{t, D}) = \log\left(\frac{\text{Total no. of documents}}{\text{No. of documents with term t in it}}\right) \quad (2)$$

Thus, tf-idf is calculated as given in equation (3)

$$\text{tf} - \text{idf}\,(\text{t, d, D}) = \text{tf}\,(\text{t, d}) * \text{idf}\,(\text{t, D}) \quad (3)$$

where t denotes the terms; d denotes each document and D denotes the collection of documents.

Feature selection is done to reduce the size of problem for learning algorithms which may improve classification accuracy due to reduction in computation requirement. This also increases the speed of classification task as the size of data to train the classifier is reduced [23], [24]. Many feature selection algorithms are available across pertinent literature. Swarm intelligence (SI) algorithms have proven capabilities of computational intelligence for dealing with the complex real world problems [6]. They often utilize the guidance of nature to search for the optimal solution. They are a class of nature inspired meta-heuristics and are population based algorithms. They are based on the hunting, breeding, etc. behaviors of birds, insects and the like. Ant colony optimization, cuckoo search, particle swarm optimization are examples of some popular swarm intelligence algorithms.

[6]Google Web Scraper: https://chrome.google.com/webstore/detail/webscraper.

[7]University of Glasgow stop-word list: http://ir.dcs.gla.ac.uk/resources/linguistic_utils/

Motivated by the adaptive learning and collective intelligence behaviors of SI algorithms, in this work, the use of particle swarm optimization (PSO) for enhancing fault classification accuracy is demonstrated. PSO is an evolutionary computation algorithm which aims to find a globally optimized solution. It was developed in 1995 by Kennedy and Eberhart [25], [26] based on the imitating social behaviors and random movements similar to that of a flock of birds or a school of fish. The original version of PSO was proposed by modifying these initial imitations. Later, inertia weight was introduced by Shi and Eberhart [27] to illustrate the standard PSO algorithm. Initially, the algorithm is flooded with a population of random solutions. These random solutions are called 'particles'. Each particle is a point in an S-dimensional search space. The i^{th} particle is represented as $X_i = (X_{i1}, X_{i2}, X_{i3}, \ldots.., X_{iS})$. Every particle has a memory to store its own best position denoted by ''pbest'' (personal best). The best experience of any particle i at a particular time t is denoted by P_i (t). The best previous position (pbest, the position giving the best fitness value) of any particle is represented as $P_i = (P_{i1}, P_{i2}, P_{i3}, \ldots\ldots P_{iS})$ and is recorded by the particles in their memory. The index of particle with the best position among all the particles in the population is represented by the symbol 'gbest' (global best). At any moment t the global best experience of swarm is denoted by g(t). So, at any point of time, we have pbest for every particle and gbest for complete swarm of particles. The concept of a flying particle is illustrated in figure 4.

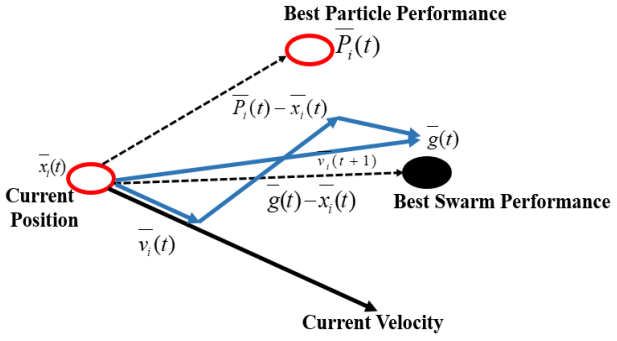

FIGURE 4. The concept of flying particle.

Some considerations are necessary to fulfill the equations for the standard PSO model. These are as follows:

- r_1 and r_2 are uniformly distributed random functions, and r_1 and $r_2 \varepsilon$ [0, 1].
- d = 1, 2, 3, ……, S is the dimension.
- The term $w.V_{id}$ (t) is the inertia term, and the coefficient w is the inertia coefficient or the inertia weight. This term provides the particles a memory capability for the exploration of new positions in the search space while flying. The original version of PSO proposed in 1995 [25], [26] did not have any inertia term. However, in 1998 Shi and Eberhart [27] added this term to formulate the standard PSO model.
- c_1 and c_2 are the acceleration coefficients.
- The term $r_1.c_1.(P_{id}(t) - X_{id}(t))$ is called cognitive component and it represents private thinking of an individual particle.
- The term $r_2.c_2.(g_d(t) - X_{id}(t))$ is called social component and it represents the collaboration among the particles.
- The two components, cognitive component and social component pull each particle towards pbest and gbest positions respectively.
- The three components, inertia term, cognitive component, and social component are combined to create a new velocity vector V_{id} (t + 1).
- This new velocity vector translates a particle position to a new updated position in the search space X_{id} (t + 1) which is probably a better location for the particle i.

This way all particles are cooperating to find out the best solution for an optimization problem. These rules are guaranteed to be obeyed by all the particles of the swarm. After a particle flies toward a new position, the performance of particle is measured according to a pre-defined fitness function. A maximum velocity, denoted by Vmax is use to limit particle's velocity on each dimension. The length of steps taken by particle through the solution space in each iteration is determined by Vmax. Small value of Vmax may lead to less exploration beyond locally good regions causing the algorithm to move towards the target slowly and particles could become trapped in local optima, whereas, if greater value has been taken for Vmax particles in the swarm will move faster towards the global optimum as they are able to move with bigger step in each iteration. Under such circumstances particles might fly past good solutions. The pseudo code for standard PSO is given in the following figure 5.

D. SUPERVISED LEARNING ALGORITHMS

Five supervised learning algorithms, namely, naïve Bayesian, decision tree, support vector machine, K-nearest neighbors and multi-layer perceptron (neural network) have been implemented and analyzed to find the best predictive learning model for fault classification. The objective is to analyze the bug report data and negative reviews and classify it into seven pre-defined categories as given by Sampath *et al.* [7]. The following table 2 gives the description of these fault types in application code.

A total of 4577 bugs were scrutinized from the five web-apps as given in table 1. The description of the classification algorithms used in this work is available across pertinent literature on machine learning [28], [29]. The training: test data split was 80:20 with a 10-fold cross validation.

Python 2.7 was used for implementation of the work. The experimentation used open source Python library, Natural Language Toolkit (NLTK)[8] for the natural language processing tasks. Open source libraries Tensorflow and Keras were used to build the neural network classifier. Other classifiers were implemented through Scikit-Learn library.

[8]Natural Language Toolkit: https://www.nltk.org

FOR each particle i
FOR each dimension d
Initialize position X_{id} randomly within range
Initialize velocity V_{id} randomly within range
End FOR
END FOR
Iteration t = 1
DO

FOR each particle i
Calculate Fitness Value
IF Fitness $(X_{id}(t))$ > Fitness $(P_{id}(t))$
$P_{id} = X_{id}$
End IF
End FOR
Chose the particle having the best Fines value as the $g_d(t)$
FOR each particle i
FOR each dimension d
Calculate velocity according to the equation

$$V_{id}(t+1) = w.V_{id}(t) + r_1.c_1.\left(P_{id}(t) - X_{id}(t)\right) + r_2.c_2.\left(g_d(t) - X_{id}(t)\right)$$

Update particle position according to the equation

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1)$$

END FOR
END FOR
t = t + 1
WHILE maximum iterations or minimum error criteria are not attained

FIGURE 5. Pseudo code for PSO.

TABLE 2. Faults types in application codes.

Fault	Type
Data Store	Query or data manipulation fault in database
Form	Pertaining to the form fields, missing and incorrect values and validation
Logic	Flow of event or control
Link	URL's and links: deadlink etc
Appearance	Display and presentation.
Compatibility	Compatibility issues: variety and versions of browsers and operating system or any other client environment.
Miscellaneous	Installation problem, security issues, others not covered in any of the above mentioned categories.

IV. RESULTS AND DISCUSSION

The empirical analysis has been divided into four parts; (i) parameter setting for PSO (ii) feature selection results (iii) comparison of accuracy with and without feature selection (iv) reduction in time for training the classifier and building the model

A. PARAMETER SETTING FOR PSO

All the parameters were set to the best values as claimed & demonstrated by Aghdam and Heidari [30]. These values are as given in table 3.

The maximum iteration and fitness function were set to 100 and 0.95 respectively. The cognitive coefficient (c_1) and the social coefficient (c_2) were set between 0.3 and 0.4 since these two values ($c_1 + c_2$) are normally limited to 4 as given by Aghdam and Heidari [30].

TABLE 3. Parameters for PSO.

Parameter	Value
Particles	20
w	0.33
c_1	0.34
c_2	0.33

B. FEATURE SELECTION RESULTS

In the proposed work, initially 956 features were extracted using tf-idf. Feature selection was carried out using PSO to obtain the reduced feature subset. The following Table 4 depicts the number of features selected using tf-idf and tf-idf + PSO.

TABLE 4. Feature selection vs feature extraction.

Algorithm	Feature Extraction (tf-idf) #Features	Feature Selection (tf-idf+PSO) #Features	Features Selected (%)
NB	956	515	53.87
DT	956	809	84.62
SVM	956	883	92.36
K-NN	956	442	46.23
MLP	956	883	92.36

The basic feature extraction based on tf-idf filter used the same number of features, i.e., 956 for all classification algorithms. Applying PSO, the minimum number of features selected were 442 for K-NN, which is 53.77 % reduction in features. The maximum was 883 features for both SVM and NN which show only 7.64% reduction in features. The figure 6 depicts the average of feature selection.

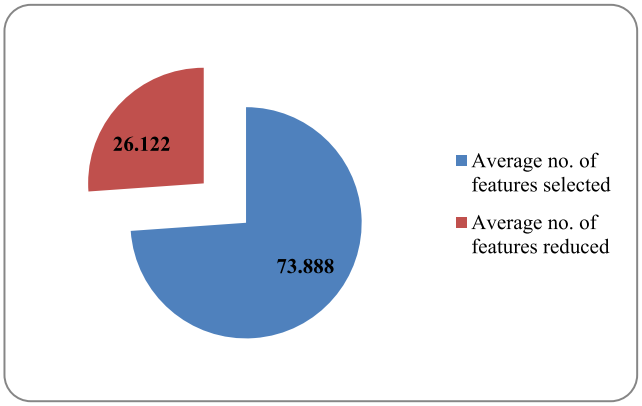

FIGURE 6. Feature reduction using PSO.

C. EFFECT OF FEATURE SELECTION ON CLASSIFICATION ACCURACY

This sub-section gives a comparison of the classification algorithms used, based on performance accuracy

(misclassification-error). It also demonstrates the advantage of using PSO based feature selection technique over the conventional tf-idf technique. Table 5 depicts the accuracy results.

TABLE 5. Accuracy obtained using before and after optimization.

Algorithm	tf-idf	tfidf + PSO	Accuracy Gain (%)
NB	75.158	88.256	13.098
DT	67.721	93.354	25.633
SVM	90.038	91.554	1.516
K-NN	79.905	90.759	10.854
MLP	89.829	92.452	2.623

The results indicate the maximum accuracy without optimization is achieved by SVM, i.e., 90.038%. The maximum accuracy gain was obtained by decision tree (25.633%). It can be clearly observed the classification performance is improved with the feature subset using PSO. The average improvement of 10.74% has been observed using PSO. Figure 7 shows the accuracy comparison graphically.

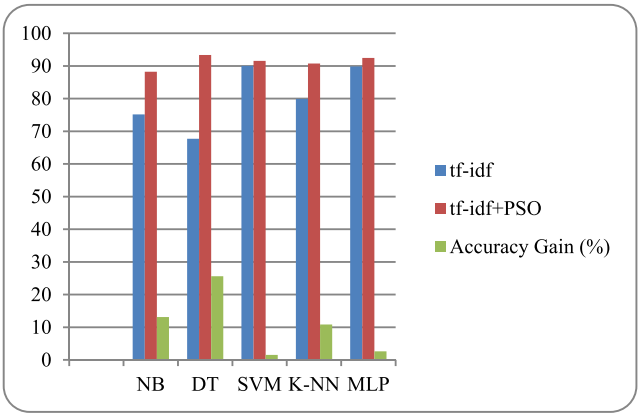

FIGURE 7. Comparison of accuracy with and without optimization.

The gain is accuracy is graphically depicted by figure 8.

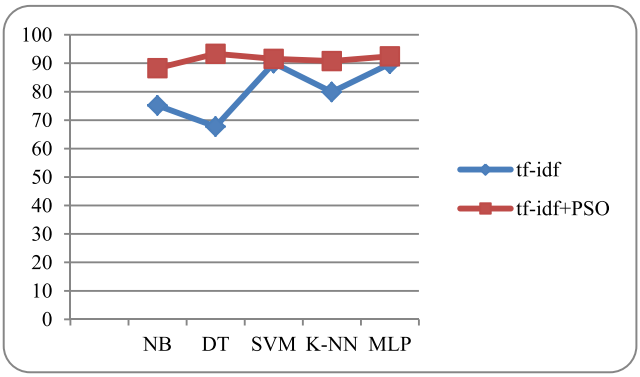

FIGURE 8. Accuracy Gain.

The performance of the training models is further evaluated using sensitivity (true positive recognition rate) and specificity (the true negative recognition rate). The following figure 9 depicts the result. DT has the highest true positive recognition rate of 98.9%.

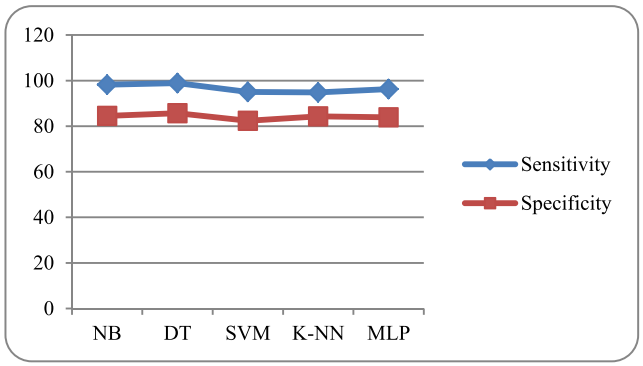

FIGURE 9. Sensitivity and Specificity of training models.

D. REDUCTION IN TIME

The aim of feature selection and optimization methods is to reduce the computational time and complexity of the prediction model. After feature selection, the five baseline classifiers are trained with the reduced feature subset and comparative analysis of the time for building the models has been done in order to evaluate reduction in time and to evaluate the effectiveness of the feature selection approach. The following figure 10 gives comparative results of the time for building various classification models thus evaluating the reduction in time.

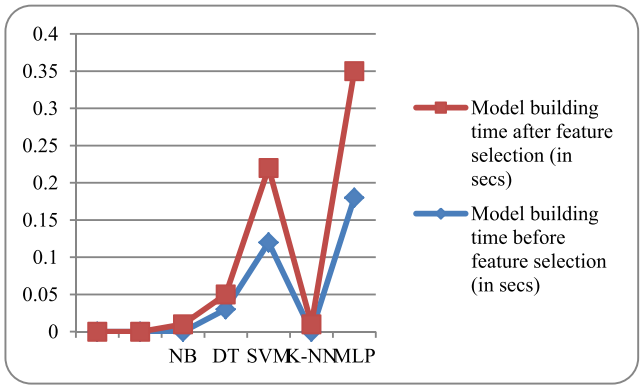

FIGURE 10. Reduction in model building time.

Maximum reduction in time is observed in building SVM with the value of 0.02 seconds whereas in all other classification models a reduction of 0.01 seconds is observed.

V. CONCLUSION

The effectiveness of fault-based testing depends on the quality of the fault model and analyzing faults in previous designs, to predict and avert similar faults in future product designs. This research proposed a predictive learning model to classify

faults which assists assessment of probable faults thus enhancing the quality of the web-apps. An empirical study to classify faults in bug reports of three open-source web applications and reviews of two play store applications using five baseline classifiers was conducted. These baselines were initially trained using conventional tf-idf feature extraction method and subsequently using an optimal feature selection method, particle swarm optimization algorithm. A total of 4577 bugs were analyzed based on accuracy as the performance metric of the classifier. The average accuracy gain is of about 11% was observed with nearly 74% features were selected on average. The empirical analysis validates that the PSO algorithm for feature selection optimization in fault classification task outperforms the elementary classification task based on feature extraction.

The results demonstrate and motivate to explore the use of other meta-heuristic algorithms, such as elephant search, bacterial foraging, cuckoo search, firefly algorithm and wolf search algorithms etc. The results can also be analyzed by using other intrinsic filters such as, information gain, chi-square and their hybrids with swarm-based wrapper algorithms. Fuzzy-logic and evolutionary algorithms can also be investigated to built optimal and robust predictive learning models for fault classification in web-apps.

REFERENCES

[1] A. Kumar and R. Goel, "Event driven test case selection for regression testing Web applications," in *Proc. IEEE-Int. Conf. Adv. Eng., Sci. Manage.*, Mar. 2012, pp. 121–127.
[2] A. Kumar and D. Gupta, "Paradigm shift from conventional software quality models to Web based quality models," *Int. J. Hybrid Intell. Syst.*, vol. 14, no. 3, pp. 167–179, 2017.
[3] J. Tian, "Quality assurance alternatives and techniques: A defect-based survey and analysis," SQP, ASQ, Dept. Comput. Sci. Eng., Southern Methodist Univ., Dallas, TX, USA, 2001, vol. 3, no. 3.
[4] Y. Guo and S. Sampath, "Web application fault classification-an exploratory study," in *Proc. 2nd ACM-IEEE Int. Symp. Empirical Softw. Eng. Meas. (ESEM)*, New York, NY, USA, 2008, pp. 303–305.
[5] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Elect. Eng.*, vol. 40, no. 1, pp. 16–28, Jan. 2014.
[6] A. Kumar, R. Khorwal, and S. Chaudhary, "A survey on sentiment analysis using swarm intelligence," *Indian J. Sci. Technol.*, vol. 9, no. 39, Oct. 2016, doi: 10.17485/ijst/2016/v9i39/100766.
[7] S. Sampath, S. Sprenkle, E. Gibson, L. Pollock, and A. S. Greenwald, "Applying concept analysis to user-session-based testing of Web applications," *IEEE Trans. Softw. Eng.*, vol. 33, no. 10, pp. 643–658, Oct. 2007.
[8] S. Elbaum, G. Rothermel, S. Karre, and M. Fisher, II, "Leveraging user-session data to support Web application testing," *IEEE Trans. Softw. Eng.*, vol. 31, no. 3, pp. 187–202, Mar. 2005.
[9] M. Li and J. Tian, "Web error classification and analysis for reliability improvement," *J. Syst. Softw.*, vol. 80, no. 6, pp. 795–804, 2007.
[10] A. Kumar, R. Chugh, R. Girdhar, and S. Aggarwal, "Classification of faults in Web applications using machine learning," in *Proc. ACM Int. Conf. Intell. Syst., Metaheuristics Swarm Intell.*, 2017, pp. 62–67.
[11] T. Mantere and J. T. Alander, "Evolutionary software engineering, A review," *Appl. Soft Comput.*, vol. 5, no. 3, pp. 315–331, 2005.
[12] F. A. Nielsen, "A new ANEW: Evaluation of a word list for sentiment analysis in microblogs," in *Proc. ESWC*, 2011, pp. 93–98.
[13] A. Kumar and T. M. Sebastian, "Sentiment analysis: A perspective on its past, present and future," *Int. J. Intell. Syst. Appl.*, vol. 4, no. 10, 1–14, 2012.
[14] A. Kumar and T. M. Sebastian, "Sentiment analysis on twitter," *Int. J. Comput. Sci. Issues*, vol. 9, no. 4, p. 372, 2012.
[15] A. Kumar and A. Sharma, "Socio-Sentic framework for sustainable agricultural governance," *Sustain. Comput., Inform. Syst.*, to be published. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210537918302336, doi: 10.1016/j.suscom.2018.08.006.
[16] A. Kumar, H. Ahuja, N. K. Singh, D. Gupta, A. Khanna, and J. J. P. C. Rodrigues, "Supported matrix factorization using distributed representations for personalised recommendations on twitter," *Comput. Elect. Eng.*, vol. 71, pp. 569–577, Oct. 2018.
[17] A. Kumar, V. Dabas, and P. Hooda, "Text classification algorithms for mining unstructured data: A SWOT analysis," in *International Journal of Information Technology*. Delhi, India: Springer, 2018, pp. 1–11.
[18] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: A statistical framework," *Int. J. Mach. Learn. Cybern.*, vol. 1, nos. 1–4, pp. 43–52, 2010.
[19] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980. [Online]. Available: https://tartarus.org/martin/PorterStemmer/
[20] M. P. S. Bhatia and A. K. Khalid, "Information retrieval and machine learning: Supporting technologies for Web mining research and practice," *Webology*, vol. 5, no. 2, p. 5, 2008.
[21] M. P. S. Bhatia and A. Kumar, "A primer on the Web information retrieval paradigm," *J. Theor. Appl. Inf. Technol.*, vol. 4, no. 7, pp. 657–662, 2008.
[22] A. Kumar, A. Jaiswal, S. Garg, S. Verma, and S. Kumar, "Sentiment analysis using cuckoo search for optimized feature selection on Kaggle tweets," *Int. J. Inf. Retr. Res.* vol. 9, no. 1, pp. 1–15, 2019.
[23] N. Omar, F. Jusoh, R. Ibrahim, and M. S. Othman, "Review of feature selection for solving classification problems," *J. Inf. Syst. Res. Innov.*, vol. 3, pp. 64–70, Feb. 2013.
[24] X. Wang, J. Yang, and X. Teng, "Feature selection based on rough sets and particle swarm optimization," *Pattern Recognit. Lett.*, vol. 28, no. 4, pp. 459–471, 2007.
[25] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, WA, Australia, vol. 4, Nov./Dec. 1995, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.
[26] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, Nagoya, Japan, Oct. 1995, pp. 39–43.
[27] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Int. Conf. Evol. Comput.*, Anchorage, AK, USA, May 1998, pp. 69–73.
[28] A. Kumar and A. Jaiswal, "Empirical study of twitter and tumblr for sentiment analysis using soft computing techniques," in *Proc. World Congr. Eng. Comput. Sci.*, vol. 1, 2017, pp. 1–5.
[29] J. Alzubi, A. Nayyar, and A. Kumar, "Machine learning from theory to algorithms: An overview," in *Proc. 2nd Nat. Conf. Comput. Intell. (NCCI)*, 2018, vol. 1142, no. 1, p. 012012.
[30] M. H. Aghdam and S. Heidari, "Feature selection using particle swarm optimization in text categorization," *J. Artif. Intell. Soft Comput. Res.*, vol. 5, no. 4, pp. 231–238, 2015.

DEEPAK KUMAR JAIN received the Bachelor of Engineering degree from Rajiv Gandhi Proudyogiki Vishwavidyalaya, India, in 2010, the Master of Technology degree from the Jaypee University of Engineering and Technology, India, in 2012, and the Ph.D. degree from the Institute of Automation, University of Chinese Academy of Sciences, Beijing, China. He is currently an Assistant Professor with the Chongqing University of Posts and Telecommunications, Chongqing, China. He has presented several papers in peer-reviewed conferences and has published numerous studies in science cited journals. His research interests include deep learning, machine learning, pattern recognition, and computer vision.

AKSHI KUMAR received the B.E. degree (Hons.) in computer science and engineering from Maharshi Dayanand University, Rohtak, in 2003, the M.Tech. degree (Hons.) in computer science and engineering from Guru Gobind Singh Indraprastha University, New Delhi, in 2005, and the Ph.D. degree in computer engineering in the area of web mining from the Faculty of Technology, University of Delhi, in 2011. She is currently an Assistant Professor with the Department of Computer Science and Engineering, Delhi Technological University (formerly Delhi College of Engineering). She has been with the university for the past 10 years. Her research interests include intelligent systems, user-generated big-data, social media analytics, and soft computing.

SAURABH RAJ SANGWAN received the bachelor's degree in computer science and engineering from DCRUST, Murthal, India, and the M.Tech. degree in software engineering from the Department of Computer Science & Engineering, Delhi Technological University, New Delhi, India, where he is currently a Research Scholar with the Web Research Group. His current research interests include intelligent systems, text mining, and social web.

GIA NHU NGUYEN received the Ph.D. degree in computer science from the Hanoi University of Science at Vietnam National University, Vietnam. He is currently the Dean of the Graduate School, Duy Tan University, Vietnam. He has a total academic teaching experience of 18 years with more than 50 publications in reputed international conferences, journals, and online book chapter contributions (indexed by: SCI, SCIE, SSCI, Scopus, and DBLP). His current research interests include network communication, security and vulnerability, network performance analysis and simulation, cloud computing, and image processing in biomedical. He is currently an Associate Editor of the *International Journal of Synthetic Emotions*.

PRAYAG TIWARI received the M.S. degree from NUST MISIS, Moscow. He is currently pursuing the Ph.D. degree with the University of Padova, Italy, where he is also a Marie Sklodowska-Curie Researcher. He was a Research Assistant with NUST MISIS and has teaching and industrial work experience. He has several publications in journals, book series, and conferences of the IEEE, ACM, Springer, Elsevier, MDPI, Taylor & Francis, and IGI-Global. His current research interests include machine learning, deep learning, quantum machine learning, and information retrieval.