Particle Swarm Optimization (PSO)-based Feature Selection: An Approach for Improving Performance of Classification Model

OLUYEMISI ADENIKE OYEDEMI [1]
IYABO OLUKEMI AWOYELU [2]
EMMANUEL ROTIMI ADAGUNODO [3]

Department of Computer Science, University of Ilesa, Ilesa, Osun State, Nigeria [1]

Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Osun State, Nigeria [2,3]

[1]yemisi.oyedemi@unilesa.edu.ng

[2] iawoyelu@oauife.edu.ng

[3] eadagun@oauife.edu.ng

Abstract. The Particle Swarm Optimization (PSO) algorithm was implemented in this paper for feature selection in order to improve the performance of web phishing classification model. The algorithm initialized a population of particles, each of which represented a possible feature subset, and then iteratively investigated subsets of features. The particles are updated according to their global and personal best locations in an effort to choose the best feature subset. A total of 36 features were selected out of the downloaded dataset comprising of 48 features. The performance of the whole dataset was measured against the performance of the selected dataset. It was observed that the performance of the classifier improved across all the metrics. The accuracy, precision, recall and F1-score increased from 93% to 96%, 92% to 97%, 92% to 96% and 93% to 97% respectively. The results indicated that the PSO algorithm effectively identified a subset of features that improved the performance of the classifier. The PSO algorithm's optimal feature subset produces superior classification performance when compared to using the whole dataset. Through feature selection, this method demonstrated how well PSO works to improve classifier accuracy and efficiency across a range of classification tasks.

Keywords: Classifier Performance, Feature Selection, Machine Learning, Particle Swarm Optimization (PSO), Optimization, Redundancy

1 Introduction

Feature selection is a process of selecting a subset of relevant features from a large number of original features to achieve better classification performance, reduce the running time of the learning algorithm and improve computation efficiency without decreasing the prediction accuracy. Feature selection is useful in cases when the dataset dimensionality is large (where the numbers of attributes is numerous) which normally makes the search space wide and creates difficulties for the data mining algorithm [1]. The data owner determines the subset of features that serves as a good sample of the entire data and this subset of features when mined generates similar performances to the entire dataset. The rationale behind feature selection is to reduce the search space by

removing irrelevant variables so that only interrelated variables with the class label are selected and used for the training [2].

Feature selection (FS) preprocesses the input dataset to assess the available attributes and keep the relevant ones because redundant and irrelevant features will reduce the classification accuracy [5]. It also helps in solving the data over fitting problem by getting rid of noisy features, shortening the running time, reducing the computational load, increasing the overall classification performance of the learning models and/or simplifying the structure of the learned classifier or models [1]. Some of the most popular metaheuristic algorithms that have been used for feature selection problems are Genetic Algorithm (GA) [3], Particle Swarm Optimization (PSO) [8], Artificial Bee Colony (ABC) and Ant Colony Optimization (ACO) [4], algorithms inspired by fish schools [9] etc.

Particle Swarm Optimization, proposed in this paper, is one of the latest evolutionary techniques. It is a stochastic, population-based computer algorithm modelled on Swarm intelligence, that gets a solution to an optimization problem in a search place, or model and predict social behavior in the presence of objectives. Swarm intelligence is based on social-psychological principles and provides insights into social behaviour, as well as contributing to engineering applications. The computation technique is inspired by social behavior such as birds flocking and fish schooling. The PSO algorithm is easier to implement and converges more quickly [6]. It is very efficient and has demonstrated high potential in handling optimization problems because of its capability for both global and intensive local searches. The choice of this algorithm is due to its best performance in terms of the outcome of the selected dataset, reliability of its efficiency, the time taken for the entire optimization process with limited number of iterations, and improved classification accuracy [10]. This reduces the complexity of the algorithms involved in the classification and the computational cost because of its capability to automatically search good features [7].

The process of feature selection consists of five basic steps as shown in Figure 1. A feature selection algorithm starts with an initialization procedure based on all the available features. This was followed by a subset discovery procedure that generates candidate subset. This procedure can start with no features, all features, and a random subset of features [2]. An evaluation function known as subset evaluation is used to measure the goodness of the generated feature subsets. This algorithm will stop according to the stopping criterion which can be based on the generation procedure or the evaluation function. The former can be a predefined number of features selected and a predetermined maximum number of iterations reached.

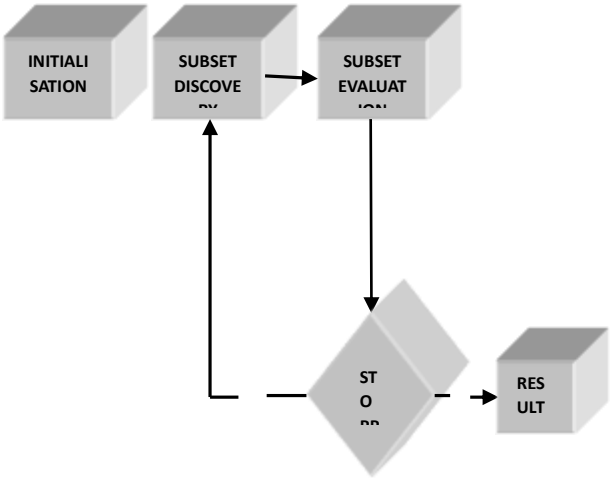

Figure 1: Feature Selection Process

The latter includes whether an optimal feature subset according to a certain evaluation function is obtained or whether addition or deletion of any feature does not produce a better subset. Finally, the validity

of the selected subset is tested by carrying out tests on unseen data. Previous studies have demonstrated the potential of PSO in various feature selection tasks. [14] showed that PSO-based feature selection could effectively reduce the number of features while maintaining or improving classification accuracy. Similarly, [13] provided a comprehensive review of PSO applications in feature selection, highlighting its adaptability to different types of datasets and classification problems.

The rest of the paper is organized as follows: Section 2 presents related works. Section 3 discusses the methodology. Section 4 presents the results and discussion. Section 5 presents the concluding remarks.

2 Related Works

A comprehensive review of related works on feature selection is presented in this section. Feature selection is a fundamental step in the machine learning pipeline, particularly for high-dimensional datasets where the presence of irrelevant or redundant features can significantly impact the performance of classifiers. Traditional statistical techniques to more sophisticated metaheuristic techniques have been developed to address this issue. In recent years, metaheuristic algorithms, including Particle Swarm Optimization (PSO), Genetic Algorithms (GA), and Ant Colony Optimization (ACO), have gained popularity for their ability to efficiently search for optimal feature subsets in large and complex search spaces. Metaheuristic algorithms have emerged as powerful tools for feature selection due to their ability to navigate complex search spaces and avoid local optimal. Genetic Algorithms (GA) [15] and Ant Colony Optimization (ACO).

[16] are among the earliest and most widely studied metaheuristics in this domain. GA, inspired by the process of natural selection, employs crossover and mutation operators to evolve feature subsets over generations. ACO, based on the foraging behavior of ants, uses pheromone trails to guide the search for optimal solutions. Both algorithms have been successfully applied to feature selection, but they often require careful tuning of parameters and can be computationally intensive. PSO, introduced by Kennedy and Eberhart in 1995 [11], has gained significant attention as a feature selection technique due to its simplicity and efficiency. Unlike GA and ACO, PSO does not rely on complex operators like crossover or pheromone updating. Instead, it mimics the social behavior of swarms, where each particle adjusts its position in the search space based on its own experience and that of the swarm. This approach allows PSO to quickly converge to optimal or near-optimal solutions, making it particularly suited for high-dimensional feature selection problems.

[12] introduced a chaotic PSO for feature selection, which uses chaotic maps to enhance the diversity of the particle swarm and prevent premature convergence. [14] introduced a discrete PSO variant for feature selection in binary classification tasks, demonstrating its superiority over standard feature selection methods. The versatility of PSO has led to its application in a wide range of domains beyond feature selection, including optimization of neural network architectures [20], clustering [17], and image processing [21]. In the context of feature selection, PSO has been applied to various types of datasets, including medical [18], text [19], and bioinformatics data [22]. These applications highlight the adaptability of PSO to different problem domains and its effectiveness in improving the performance of machine learning models. PSO has proven to be a powerful and flexible tool for feature selection, offering advantages in terms of simplicity, efficiency, and adaptability. The algorithm's ability to balance exploration and

exploitation makes it well-suited for high-dimensional problems where traditional methods may fall short.

This paper applied PSO to a real-world dataset extracted from browsers of webpages and URLs and downloaded from the UCI machine learning repository. The selected dataset demonstrates high potential to build a classification model for web phishing detection.

3 Methodology

This section outlines the process of implementing the Particle Swarm Optimization (PSO) algorithm to select relevant features from the dataset, as well as the experimental setup used to evaluate its effectiveness. The goal at this phase was to obtain the relevant feature values for each instance of the training corpus which can be used by the web phishing classification model. Classifiers tend to degrade in performance when faced with too many features that are not very necessary to predict. This problem is therefore viewed as an optimization problem whose objective is to select the most relevant features to improve the predictive accuracy of the classification model.

The dataset used in this study is extracted from browsers of webpages and URLs, and was downloaded from UCI machine learning repository at http://dx.doi.org/10.17632/h3cgnj8hft.1. The dataset consists of 10,000 instances of 5000 phishing web pages and 5000 legitimate web pages, large number of websites comprising 48 features. The features includes attributes such as NumDots, SubdomainLevel, PathLevel, URLLength, NumDash, AtSymbol, TildeSymbol, NumUnderscore, NumPercent, , NumAmpersand, NumHash, NumNumericChars, NoHttps, RandomString, IPAddress, , DomainInPaths, HttpsInHostname, HostnameLength, PathLength, QueryLength, DoubleSlashInPath,, PctExtResourceURLs, and ExtFavicon.

The dataset was checked for missing values, outliers, and inconsistencies. Missing values were handled by imputation using the mean or median for continuous variables, or by removing samples with missing data for categorical variables. Outliers were identified and either transformed or removed based on their impact on the overall distribution. To ensure that all features contributed equally to the analysis, the continuous features were normalized using min-max scaling, which transformed the feature values to a range between 0 and 1. This step was crucial for the PSO algorithm, as it relies on distance metrics for evaluating feature subsets. Feature normalization changes all the features to the same scale to allow for faster convergence on learning and more uniform influence for all weights. The dataset was then split into training and testing sets using a standard 80-20 split. The training set was used to evaluate the performance of different feature subsets during the optimization process, while the testing set was reserved for final evaluation of the selected feature subset.

3.1 Implementation of Particle Swarm Optimization (PSO) for feature selection

In order to remove the non-relevant features from the dataset, Particle Swarm Optimizer was used to do feature selection on the dataset. The PSO algorithm was implemented in a Jupyter Notebook using Python 3.7.10. The PSO-based Feature Selection flowchart is shown in Figure 2 while the Particle Swarm Algorithm is shown in Table 1. In Particle Swarm Optimization process, each potential solution is called a particle, which are the 48 features of the dataset, with no weight and no volume. In order to determine whether a feature will be selected or not, a threshold is needed to compare with the value of the vector [23].

The algorithm works by having a population (called a swarm) of candidate solutions

(called particles). Each particle, located in the hyperspace has random position φ_i and velocity ϑ_i. The basic update rule for the position and the speed is depicted in Equation 1 and Equation.2 respectively.

$$\varphi_i(t+1) = \varphi_i + \vartheta_i(t+1) \quad (1)$$

$$v_i(t+1) = \omega\vartheta_i(t) + c_1 r_1(p_i - x_i) + c_2 r_2(g - x_i) \quad (2)$$

Where ω denotes inertia weight that controls the balance between exploitation and exploration. A large inertia weight improves the global search ability while a small inertia weight enhances local search ability. c_1 and c_2 denotes cognitive and social learning factors, respectively, r_1 and r_2 represent two independent uniform random numbers with values between 0 and 1, $p_i$is personal best position of particle i, and finally, g is global best position among all particles in the swarm. The particle swarm algorithm begins by creating the initial particles, and assigning them initial velocities at time *t*. These particles are moved around in the search space according to a few simple formulae. The movements of the particles are guided by their own best known position in the search space as well as the entire swarm's best known position. When improved positions are being discovered these will then come to guide the movements of the swarm. It evaluates the objective function at each particle location, and determines the best (lowest) function value and the best location. It chooses new velocities at time *t*+1, based on the current velocity, the particles' individual best locations, and the best locations of their neighbours. It then iteratively updates the particle locations (the new location is the old one plus the velocity, modified to keep particles within bounds), velocities, and neighbors. Iterations proceed until the algorithm reaches a stopping criterion.

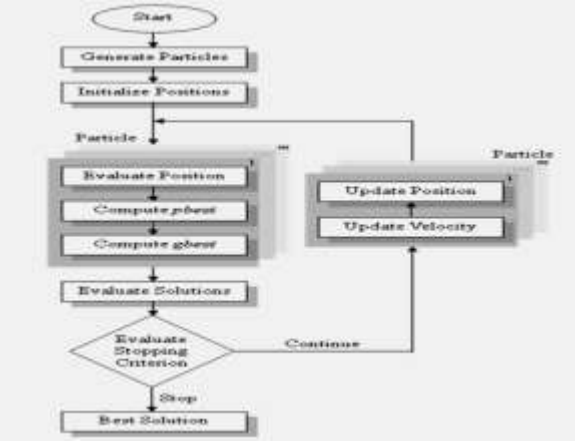

Figure 2 PSO-based Feature Selection Flowchart

Let *S* be the number of particles in the swarm, each having a position $\mathbf{x}_i \in \mathbb{R}^n$ in the search-space and a velocity $\mathbf{v}_i \in \mathbb{R}^n$. Let p_i be the best known position of particle *i* and let g be the best known position of the entire swarm as shown in Table 1 above. The values b_{low} and b_{up} represents the lower and upper boundaries of the search space. The termination criterion is the number of iterations performed, or a solution where the adequate objective function value is found. The parameters ω, φ_p, and φ_g are selected by the practitioner and control the behaviour and efficacy of the PSO method. All of particles have fitness values which are evaluated by the fitness function as shown in Equation 3.

$$\text{Fitness}_i = w_1.\text{acc}_i + w_2.1 - \frac{\sum_{i=1}^{|F|} fi}{Nf} \quad (3)$$

Where w_1 and w_2 are random variables selected between 0 and 1, acc_i is the accuracy of the selected features in the ith iteration, f_i is the flag value 1 or 0. "1" represents selected feature, "0" represents non-selected feature. F and N_f represents the features and number of features respectively.

Table 1: Particle Swarm Optimisation Algorithm

Basic Particle Swarm Optimisation Based Feature Selection Algorithm

for *each particle i = 1,…S* ***do***
Initialize the particle's position with a uniform distributed random vector: X ~U (b_{low}, b_{up})
Initialize the particle's best known position to its initial position: P_i x_i
If $f(p_i) < f(g)$ ***then***
Update the swarm's best known position: g p_i
Initialize the particle's velocity: $V_i \sim U(-|b_{up} - b_{low}|, |b_{up} - b_{low}|$
While *a terminator criterion is not met* ***do****:*
for *each particle I = 1,…S* ***do***
for *each dimension d = 1,…n* ***do***
Pick random numbers: $r_p, r_g \sim U(0,1)$
Update the particle's velocity: $V_{i,d} \leftarrow \omega \mathbf{v}_{i,d} + \varphi_p r_p (\mathbf{p}_{i,d} - \mathbf{x}_{i,d}) + \varphi_g r_g (\mathbf{g}_d - \mathbf{x}_{i,d})$
Update the particle's position: $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i$
if $f(x_i) < f(p_i)$ ***then***
Update the particle's best known position: $\mathbf{p}_i \leftarrow \mathbf{x}_i$
if $f(p_i) < f(g)$ ***then***
Update the swarm's best known position: $\mathbf{g} \leftarrow \mathbf{p}$

Parameters added to the algorithm were number of processes and iterations which were set to 10 and 100 respectively. The result generated after the optimizer ran for twenty-four (24) hours selected thirty - six (36) features.

3.2 Performance Evaluation

Once the PSO algorithm identified the optimal feature subset, the selected features were used to train a Random Forest classifier. Random forest (RF) is a machine learning classifier that is well-suited for the processing of high dimensional datasets, due to its robustness, versatility, and high universality. A random sample of rows from the training data will be taken for each tree in order for a tree to be grown. From the sample taken, a subset of features will be taken to be used for splitting on each tree, each tree will be grown to the largest extent specified by the parameters until it reaches a vote for the class. The performance of the classifier using the selected features was compared to that using the full feature set. The comparison was based on the same metrics used during fitness evaluation: accuracy, precision, recall, and F1 score. The performance of the PSO-based feature selection was evaluated on the test set, which was not used during the optimization process. The classifier's performance on the test set with the selected features was compared to its performance with all features to demonstrate the impact of feature selection and its effect on computational efficiency.

3.3 Experimental Setup

The experiments were conducted on a standard computing environment with a Python-based software stack, including libraries such as Pandas, Numpy, and Scikit-learn. The Jupyter Notebook environment was used for development and experimentation, providing an interactive platform for coding, visualizing results, and documenting the process. To ensure the effectiveness of the PSO algorithm, key hyperparameters such as swarm size, inertia weight, cognitive coefficient, and social coefficient were tuned through grid search. The hyperparameters were selected based on their ability to balance exploration and exploitation, ensuring that the swarm could efficiently search the feature space without getting trapped in local optima. Finally, the results of the PSO-based feature selection were compared with baseline feature selection methods. This comparison provided

a benchmark to assess the advantages of the PSO algorithm in terms of feature subset quality and classifier performance.

4 Results and Discussion

This section presents the results obtained from the Particle Swarm Optimization (PSO) algorithm and the discussion of the findings. The performance of the classifier is evaluated using both the complete dataset and the selected dataset by the PSO algorithm. The comparison focuses on key performance metrics, including accuracy, precision, recall, and F1 score, as well as the dimensionality reduction achieved through feature selection. This is shown in the table 2 below. The table shows a significant improvement in the performance of the selected datasets. A total of 36 features were selected out of the downloaded dataset comprising of 48 features. The performance of the classifier improved across all the metrics. The accuracy, precision, recall and F1-score increased from 93% to 96%, 92% to 97%, 92% to 96% and 93% to 97% respectively. The results indicated that the PSO algorithm effectively identified a subset of features that improved the performance of the classifier.

Table 2: Performance evaluation comparison of whole and Selected Features

Metrics (%)	Complete Dataset	Selected Features
Accuracy	93.00	96.00
Precision	92.00	97.00
Recall	92.00	96.00
F1 - Score	93.00	97.00

The robustness of the PSO algorithm in handling high-dimensional feature spaces is evident from the results. The improvements in classifier performance achieved through feature selection underscore the importance of identifying relevant features. By eliminating irrelevant or redundant features, the classifier is able to focus on the most informative aspects of the data, leading to better generalization on unseen data. This is reflected in the improved accuracy, precision, recall, and F1 score. The reduction in the number of features not only improves the classifier's performance but also reduces the computational resources required for training. This is particularly important in real-world applications such as web phishing detection, healthcare applications, etc. where computational efficiency is a critical factor. The PSO algorithm's ability to achieve significant dimensionality reduction without compromising performance makes it a valuable tool for feature selection in high-dimensional datasets.

5. Conclusion

The PSO algorithm demonstrated significant potential in enhancing classifier performance through effective feature selection. Its application in this study serves as a proof of concept for its broader use in various classification tasks, especially in scenarios involving high-dimensional datasets. The findings contribute to the growing body of research on metaheuristic algorithms for feature selection and highlight the ongoing need for innovative approaches to optimize machine learning models in complex domains.

6.0 References

[1] Gheyas, et al. Feature subset selection in large dimensionality domains," *Pattern Recognition*, 43(1): 5–13, 2010.

[2] Thabtah et.al. Prediction phase in associative classification mining, *International Journal of Software Engineering and Knowledge Engineering*, 21(6): 855–876, 2011.

[3] Waqas,et al. Feature subset selection using multi-objective genetic algorithms. *In 13th International Conference on IEEE International Multi-Topic Conference (INMIC)*, 1–6, 2009.

[4] Ke, et al. A multiobjective ACO algorithm for rough feature selection. *In: Second Pacific-Asia Conference on Circuits, Communications and System (PACCS)*. 1: 207–210, 2010.

[5] Dash, et al. Feature selection for classification. *Intelligent Data Analysis*. 1(4): 131– 156, 1997.

[6] Kennedy, et al. Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. In: IEEE World Congress on Computational Intelligence. 78–83., 1998.

[7] Moraglio, et al. Geometric Particle Swarm Optimisation. In: Genetic Programming. *Springer, Lecture Note in Computer Science (LNCS),* 4445: 125-136, 2007.

[8] Mohemmed, et al. Particle Swarm Optimization based Adaboost for face detection. *In: IEEE Congress on Evolutionary Computation,* 2494–2501, 2009.

[9] Wang, et al. A novel rough set reduction algorithm to feature selection based on artificial fish swarm algorithm, In *Proceedings of the International Conference in Swarm Intelligence*, Springer, 24-35, 2014.

[10] Hassan, et al. A comparison of particle swarm optimization and the genetic algorithm, In *Proceedings of the 46thAIAA/ASME/ASCE/AHS/ASC Structures, Structural Dy-namics and Materials Conference* (Austin, TX), 2005.

[11] Kennedy, J et al. Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks, 4, pages 1942-1948, 1995.

[12] Chuang, L. et al. (2011). Chaotic particle swarm optimization for feature selection in biomedical datasets. Journal of Medical and Biological Engineering, 31(1), 25-34, 2011.

[13] Xue, et al.. Particle swarm optimization for feature selection in classification: A multi-objective approach. *IEEE Transactions on Cybernetics*, 47(3), 757-770, 2016.

[14] Unler, et al.. A discrete particle swarm optimization method for feature selection in binary classification problems. *European Journal of Operational Research*, 206(3), 528-539, 2010.

[15] Holland, J. H. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. MIT press, 1992.

[16] Dorigo, M., et al. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE*

Transactions on Evolutionary Computation, 1(1), 53-66, 1997.

[17] Omran, M. et al. (2007). Particle swarm optimization method for image clustering. International *Journal of Pattern Recognition and Artificial Intelligence*, 21(03), 477-498, 2007.

[18] Ghosh, K et al. Medical image segmentation using particle swarm optimization aided clustering techniques. *Applied Soft Computing*, 26, 174-186, 2015.

[19] Li, X., & Wang, S. (2008). A hybrid particle swarm optimization method for supervised classification. International Journal of Hybrid Intelligent Systems, 5(1), 35-45.

[20] Poli, R., et al. Particle swarm optimization: An overview. Swarm Intelligence, 1(1), 33-57, 2007.

[21] Cui, X et al. Document clustering analysis based on hybrid PSO+ K-means algorithm. *Journal of Computer Sciences,* 1(1), 27-33, 2005.

[22] Rajesh, R et al. Classification of ECG beats using PSO and decision tree. *Procedia Computer Science*, 115, 327-336, 2017.

[23] Mehdi, H.et al. Feature selection using particle swarm optimization in text categorization. *Journal of Artificial Intelligence and Soft Computing Research* (JAISCR), 5(4): 231-238, 2015.