






Applying Differential Evolution with Threshold Mechanism for Feature Selection on a Phishing Websites Classification

Lucija Brezočnik^(✉) , Iztok Fister Jr. , and Grega Vrbančič 

Institute of Informatics, Faculty of Electrical Engineering and Computer Science,
University of Maribor, Koroška cesta 46, 2000 Maribor, Slovenia
{lucija.brezocnik, iztok.fister1, grega.vrbancic}@um.si

Abstract. The rapid growth of data and the need for its proper analysis still presents a big problem for intelligent data analysis and machine learning algorithms. In order to gain a better insight into the problem being analyzed, researchers today are trying to find solutions for reducing the dimensionality of the data, by adopting algorithms that could reveal the most informative features out of the data. For this purpose, in this paper we propose a novel feature selection method based on differential evolution with a threshold mechanism. The proposed method was tested on a phishing website classification problem and evaluated with two experiments. The experimental results revealed that the proposed method performed the best in all of the test cases.

Keywords: Classification · Differential evolution · Feature selection · Intelligent data analysis

1 Introduction

Big data, Blockchain, Internet-of-Things (IoT), and Data Science are just a few buzzwords that depict the modern technological world. All of them are linked with an amount of data that is rapidly increasing every day. At present, some influential people in the world say that data is the new oil. Interestingly, data is not useful unless it is explored with methods that are tailored to wards knowledge discovery.

There are currently numerous methods for knowledge discovery of data, e.g., classification, association rule mining, and clustering. Classification problems are commonly solved by machine-learning algorithms where the first step is data pre-processing. Data pre-processing is considered to be the hardest and most complex step in the whole machine learning ecosystem. Here, we are confronted with data that will play a role in the following steps of a pipeline. Sometimes data can be very ugly, such as in cases of missing data, non-standardized data, etc. whilst, sometimes even raw data that is well collected can be problematic.

But what happens if the data that we want to analyze consists of hundreds of instances where each of those instances has thousands, or tens of thousands, of features? Such high-dimensional data constitutes a serious problem for modern machine-learning algorithms because of the so-called curse of dimensionality. To overcome such a problem, it is necessary to find a way to reduce the number of features. Generally, two techniques are often used: feature selection and feature extraction. The latter creates new variables as a combination of others to reduce the dimensionality, while feature selection works by removing features that are not relevant or are redundant [1].

Feature selection (FS) can also be modeled as an optimization problem. However, the biggest problem is the significant time complexity. On the other hand, in order to tackle this problem, researchers have recently utilized some stochastic population-based nature-inspired algorithms that can find pseudo-optimal solutions in real-time [2]. There exist various feature selection methods that are based on stochastic population-based nature-inspired algorithms [3, 9, 12].

Fister et al. have recently proposed a new self-adaptive differential evolution with a threshold mechanism for feature selection. The authors introduced a new threshold mechanism which extends the basic self-adaptive differential evolution by adding another feature threshold and thus mechanically control the presence/absence of a particular feature in the solution [2]. In this way, the optimal threshold as well as the optimal features are searched for during the optimization process. Inspired by previous studies [2], here we apply a threshold mechanism in canonical differential evolution [8] as well as apply proposed methods on a phishing website classification.

Altogether, the main contributions of this paper can be summarized as follows:

- a novel differential evolution for feature selection where a threshold mechanism (DEFSTH) is proposed,
- the proposed method is evaluated on a phishing dataset, and
- the performance comparison study is conducted on the most commonly used conventional classifiers.

The structure of this paper is as follows. Section 2 outlines the fundamentals of the proposed DEFSTH method, which is later tested by the experiment presented in Sect. 3. Section 4 depicts the results, while Sect. 5 concludes the paper and outlines directions for future work.

2 The Differential Evolution for Feature Selection with a Threshold Mechanism

A proposed differential evolution with a threshold mechanism for feature selection (DEFSTH) extends the Differential Evolution (DE) algorithm with threshold mechanism and utilizes it for feature selection. The method is explained in detail in the following subsections. Subsection 2.1 comprises steps made in the initialization phase, while the evaluation process of the method is covered in Subsects. 2.2 and 2.3.

2.1 Initialization

In the initialization phase of the method, the following parameters are set: the lower (*Lower*) and upper (*Upper*) bounds of the search space, the threshold (*TH*), the population size (*NP*), the number of function evaluations (*nFES*), the scaling factor (*F*), the crossover rate (*CR*), and the number of folds (*k*). Those parameters together with the initialization process of individuals are presented in detail in Subsect. 2.2.

2.2 Differential Evolution with Feature Selection

The core of the method DEFSTH is explained in detail in the following two subsections. Subsection 2.2 presents the basics of the DE algorithm while Subsect. 2.2 defines the FS problem.

Fundamentals of Differential Evolution. Differential Evolution is an evolutionary algorithm used widely in solving many combinatorial, continuous, as well as real-world problems. DE was proposed by Storn and Price in 1997 [8]. A population in DE consists of individuals that are represented as real-value-coded vectors representing the candidate solutions:

$$\mathbf{x}_i^{(t)} = (x_{i,1}^{(t)}, \dots, x_{i,n}^{(t)}), \quad \text{for } i = 1, \dots, NP, \quad (1)$$

where each element of the solution is in the interval $x_{i,1}^{(t)} \in [x_i^{(L)}, x_i^{(U)}]$, and $x_i^{(L)}$ and $x_i^{(U)}$ denote the lower and upper bounds of the i -th variable. The DE is composed of three variation operators, i.e., mutation, crossover, and selection.

Mutation in DE is expressed as follows:

$$\mathbf{u}_i^{(t)} = \mathbf{x}_{r1}^{(t)} + F \cdot (\mathbf{x}_{r2}^{(t)} - \mathbf{x}_{r3}^{(t)}), \quad \text{for } i = 1, \dots, NP, \quad (2)$$

where $r1$, $r2$, $r3$ are randomly selected values in the interval $[1 \dots NP]$.

Crossover in DE is expressed as follows:

$$w_{i,j}^{(t+1)} = \begin{cases} u_{i,j}^{(t)} & \text{rand}_j(0, 1) \leq CR \vee j = j_{rand}, \\ x_{i,j}^{(t)} & \text{otherwise,} \end{cases} \quad (3)$$

Selection in DE is expressed as:

$$\mathbf{x}_i^{(t+1)} = \begin{cases} \mathbf{w}_i^{(t)} & \text{if } f(\mathbf{w}_i^{(t)}) \leq f(\mathbf{x}_i^{(t)}), \\ \mathbf{x}_i^{(t)} & \text{otherwise.} \end{cases} \quad (4)$$

Feature Selection Mechanism. In order to apply the FS mechanism into DE, some modifications of the latter were necessarily introduced.

Individuals in the DEFSTH method are represented as a vector containing real values:

$$\mathbf{x}_i^{(t)} = (x_{i,0}^{(t)}, \dots, x_{i,M}^{(t)}, TH_i^{(t)}), \quad (5)$$

for $i = 0, \dots, NP$, where each feature $x_{i,0}^{(t)}$ for $i = 0, \dots, n$ is drawn from the interval $[0, 1]$. $TH^{(t)}$ determines if the corresponding feature is present or absent in the solution. This mapping is expressed as follows:

$$a_{i,j}^{(t)} = \begin{cases} 0, & \text{if } x_{i,j}^{(t)} \leq TH^{(t)} \\ 1, & \text{otherwise,} \end{cases} \quad (6)$$

Vector \mathbf{a}_i presents a matrix, determining the presence or absence of the observed j -th feature in the i -th solution. Let us mention that value 1 means that the feature is present, while the value 0 means that the feature is absent in the solution.

There is a theoretical chance that vector a_i would have only zero values, meaning that no feature is selected. If such a marginal case occurs, the proposed method returns the maximum value of the fitness function of that individual, i.e., 1.

2.3 Fitness Function Evaluation

To evaluate the fitness value of each solution produced by DEFSTH, we utilized the Logistic Regression (LR) classifier applying the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS or LM-BFGS) [5] optimization algorithm with the following settings: Maximum iterations (*max_iter*) was set to 100, tolerance for stopping criteria to $tol = 10^{-4}$, and inverse of regularization strength $C = 1.0$.

The evaluation of the fitness function was conducted using the Logistic Regression classifier calculating the accuracy against the test subset of the initial training set and can be formally expressed as presented in Eq. (7), where *test_acc* stands for the previously mentioned calculated accuracy. Given the nature of the DE algorithm, which is basically designed to search for the global minimum, we are converting the problem of searching the best-evaluated individual using fitness function, to the problem of searching for the global minimum via the subtraction of the accuracy from a value of 1.

$$f(\text{test_acc}) = 1 - \text{test_acc} \quad (7)$$

The fitness function evaluation is performed for each produced individual until the stopping criteria, in our case, the *nFES*, is reached.

3 Experiment

The experimental approach was utilized to show the performance of the proposed DEFSTH method on a dataset presented in detail in Subsect. 3.2. How the experiment was carried out along with the evaluation method and metrics are shown in Subsect. 3.3. In the following Subsection, the used setup is explicitly listed.

The method was implemented in the Python programming language where two external libraries were used, i.e. NiaPy [10], a micro-framework for building nature-inspired algorithms, and Scikit-learn [7], a Python module integrating a vast number of machine-learning algorithms.

3.1 Setup

The experiment was conducted using a quad-core Intel Core i7-6700 CPU with a base clock speed at 3.4 GHz and 16 GB of DDR4 memory, running the Linux Mint 19 operating system.

To initialize the DE algorithm, the following optimal parameter settings were manually determined after an extensive tuning of parameters. The population size NP was set to 40, the number of function evaluations $nFES$ was set to 1000, the scaling factor F and crossover rate CR were set to 0.5 and 0.9, respectively.

3.2 Test Data

For this experiment, we composed a dataset on our own. Using the Phishtank website [6], we collected a list of 30,647 community-confirmed URLs of phishing websites and 58,000 legitimate website URLs. The legitimate URLs are gathered from a list of community-labeled and organized URLs containing the objectively reported news and top Alexa ranking websites and are thus legitimate. Using a total of 88,647 URLs, we extracted mostly address-bar based and domain-based features, extracting a total of 111 features, without the target class (phishing or legitimate).

In order to extract the previously mentioned address-bar based features, we performed a count of special characters or symbols of different parts of a URL such as the whole URL, domain part of the URL, the parameters part of the URL, etc. More information about the dataset is available at the URL [11].

3.3 Evaluation Method and Metrics

To exhaustively evaluate the performance of the proposed DEFSTH method, we conducted an evaluation using three predictive performance measures: the accuracy (ACC), $F1$ score and area under the ROC curve (AUC). The accuracy measures the ratio of correctly classified website instances regardless of their class, while the $F1$ score presents a harmonic mean of the precision and recall, averaged over all decision classes, whereas precision refers to the positive predictive value and recall refers to the true positive rate. And the aggregate measure of performance across all possible classification thresholds is presented by an AUC metric.

To objectively evaluate the performance of the proposed DEFSTH method and compare the performance against the conventional classification methods, we conducted a gold standard 10-fold cross validation [4] procedure. The 10-fold cross validation divides a given dataset into train and test sets at a ratio of 90:10.

In the same manner, the procedure is repeated for a total of 10 times, each time using a different test set for validation.

In the first experiment, we measured the performance of Logistic Regression, Naive Bayes (NB), k -Nearest-Neighbors (KNN), Decision Tree (DT) and Multilayer Perceptron (MLP) classifiers without any feature selection utilization or any kind of pre-processing. Also, while conducting the second experiment, measuring the performance of the proposed DEFSTH method, we used the same classifiers as in the first experiment to adequately compare the obtained results. All performance classifiers were initialized with the Scikit-learn [7] default settings.

All the presented results in the following section are averaged ACC, $F1$ score and AUC values, obtained on the test websites instances over all runs for each of 10 folds, if not specified otherwise.

4 Results

A 10-fold comparison of average metrics (ACC, AUC, and $F1$) with a subset of features and without a whole set of features, via the DEFSTH method, is presented in Table 1 and in Fig. 1. The results show that the proposed method obtained better results in all of the cases. After the utilization of the DEFSTH, the highest accuracy was achieved by the NB classification method (96.82%), closely followed by the KNN (96.07%). The accuracies of the remaining classification methods DT and MLP were 93.65% and 92.93%, respectively. A similar description could also be used for the results of AUC and $F1$. After a utilization of the DEFSTH algorithm, NB again performed the best in both cases, by obtaining results of 96.38% and 95.38% for AUC and $F1$, respectively.

Table 1. Comparison of average accuracies, areas under the curves and $F1$ scores, conducted 10-fold, with and without the utilization of DEFSTH.

Metrics		ACC [%]	AUC[%]	$F1$ [%]
Logistic Regression	Without FS	92.06 \pm 0.42	92.03 \pm 0.42	89.07 \pm 0.64
	DEFSTH	92.3 \pm 1.91	92.1 \pm 2.33	89.12 \pm 2.79
Naive Bayes	Without FS	92.17 \pm 0.36	91.80 \pm 0.35	88.89 \pm 0.48
	DEFSTH	96.82 \pm 0.21	96.38 \pm 0.25	95.38 \pm 0.31
k-Nearest Neighbors	Without FS	84.29 \pm 0.30	79.22 \pm 0.42	73.43 \pm 0.62
	DEFSTH	96.07 \pm 0.51	95.54 \pm 0.62	94.29 \pm 0.75
Decision Tree	Without FS	87.0 \pm 0.60	89.45 \pm 0.43	83.82 \pm 0.62
	DEFSTH	93.65 \pm 0.32	92.96 \pm 0.23	90.81 \pm 0.32
Multilayer Perceptron	Without FS	88.06 \pm 0.22	86.56 \pm 0.28	82.55 \pm 0.34
	DEFSTH	92.93 \pm 0.42	92.45 \pm 0.44	89.89 \pm 0.59

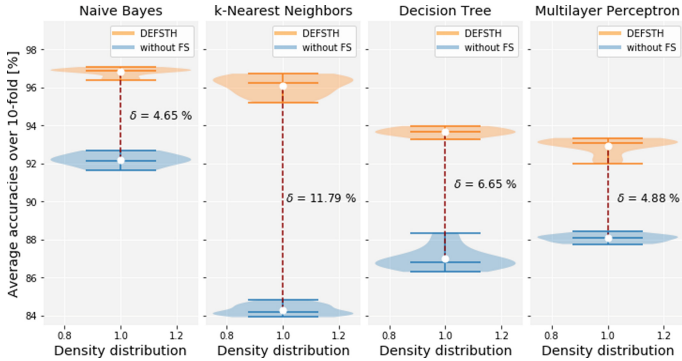


Fig. 1. Comparison of ACC over 10-fold, with and without utilization of DEFSTH.

As explained in the aforementioned results, LR was exempted since the performance of it was used to evaluate each individual (subset of features) in the process of the optimization. Thus, the results on the test data, when using DEFSTH or not, are practically the same, as expected (see row Logistic Regression in Table 1).

Since the experiment used 10-fold cross-validation, we wanted to see which features were selected in all of the folds, which in nine of the ten folds, etc. Feature 100 was chosen in all 10-folds. This feature represents the time (in days) of domain activation. Six features (3, 18, 19, 20, 40, and 83) were chosen nine times or in 5.41% of all cases. Those features represent the number of slashes (/) in the URL, URL length, number of dots (.) in the domain, number of hyphens (-) in the domain, number of dots (.) in the directory, and number of “and” signs (&) in the parameters. In eight out of ten folds, again six features were selected (23, 34, 41, 42, 47, and 106). Those features represent the number of question marks (?) in the domain, number of dollar signs (\$) in the domain, number of hyphens (-) in the directory, number of underlines (.) in the directory, number of “and” signs (&) in the directory, and is a site that has a valid TLS/SSL Certificate.

Features that were never selected are features 11 and 101, representing Autonomous system (AS) number and the time (in days) of the domain expiration, respectively.

5 Conclusion

In this work, we proposed a novel method based on DE with a threshold mechanism for FS. Two experiments were conducted using a phishing dataset, to assess its performance. The findings show that the proposed method performed best in all of the test cases and extracted highly informative features.

Based on these encouraging results, we plan to extend our research, apply the DEFSTH method to various problems including the datasets with multiple classes. Since the proposed method is implemented modularly, we plan to

extend it further and modify it with a weighted fitness function or different algorithms, such as particle swarm optimization, bat algorithm, firefly algorithm, etc. Moreover, besides applying conventional classification methods, as we did in our experiment, we also plan to utilize ensemble classification methods such as Random Forest, Bagging, Adaptive Boosting, Gradient Boosting, etc.

Acknowledgment. The authors acknowledge the financial support from the Slovenian Research Agency (Research Core Funding No. P2-0057).

References

1. Brezočnik, L., Fister, I., Podgorelec, V.: Swarm intelligence algorithms for feature selection: a review. *Appl. Sci.* **8**(9) (2018). <https://doi.org/10.3390/app8091521>
2. Fister, D., Fister, I., Jagric, T., Fister Jr., I., Brest, J.: A novel self-adaptive differential evolution for feature selection using threshold mechanism. In: *IEEE SSCI2018 Symposium Series on Computational Intelligence*, pp. 17–24 (2018)
3. Khushaba, R.N., Al-Ani, A., AlSukker, A., Al-Jumaily, A.: A combined ant colony and differential evolution feature selection algorithm. In: Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., Winfield, A.F.T. (eds.) *ANTS 2008. LNCS*, vol. 5217, pp. 1–12. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87527-7_1
4. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI 1995*, vol. 2, pp. 1137–1143. Morgan Kaufmann Publishers Inc., San Francisco (1995)
5. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. *Math. Program.* **45**(1–3), 503–528 (1989)
6. OpenDNS: PhishTank data archives. <https://www.phishtank.com/>. Accessed 21 Feb 2019
7. Pedregosa, F., et al.: Scikit-learn: machine learning in python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
8. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.* **11**(4), 341–359 (1997)
9. Unler, A., Murat, A.: A discrete particle swarm optimization method for feature selection in binary classification problems. *Eur. J. Oper. Res.* **206**(3), 528–539 (2010)
10. Vrbančič, G., Brezočnik, L., Mlakar, U., Fister, D., Fister Jr., I.: NiaPy: python microframework for building nature-inspired algorithms. *J. Open Source Softw.* **3** (2018). <https://doi.org/10.21105/joss.00613>
11. Vrbančič, G.: Phishing dataset (2019). <https://github.com/GregaVrbancic/Phishing-Dataset>. Accessed 23 May 2019
12. Zorarpacı, E., Özel, S.A.: A hybrid approach of differential evolution and artificial bee colony for feature selection. *Expert Syst. Appl.* **62**, 91–103 (2016)