



New framework for Improving Random Forest Classification Accuracy

Thamer Al-Rousan

Faculty of Information Technology, Isra University,

Amman-Jordan,

Thamer.rousan@iu.edu.jo

Received Date : January 1, 2022

Accepted Date : January 27, 2022

Published Date : February 07, 2022

ABSTRACT

Iterating over every possible combination of features and building each combination as a decision tree takes massive processing power especially when there many features to select from. The main drawback with using decision tree classifiers is the tendency of the tree to be over fitted to a specific scenario. The random forest classifier resolves this issue by using randomly selected features as nodes. The problem with this approach is that it requires more time and computational power to construct the trees. In this paper we employed an optimization algorithm called Binary Particle Swarm. The binary particle swarm optimization algorithm is a powerful algorithm in the field of optimization. We used this algorithm to pick the best features that represent a dataset as input for a random forest classifier. We have achieved impeccable results in terms of accuracy and precision while maintaining minimum user interaction. We used the Wisconsin breast cancer dataset which can be obtained from the UCI machine learning repository. In this dataset, the objective is to predict whether the passenger has survived or not based on the provided attributes. We obtained a 97% on average and a best 98% classification accuracy on the Wisconsin breast cancer dataset.

Key words : Random Forest, Binary Particle Swarm, Decision Tree, Machine Learning

1. INTRODUCTION

Random forest classifier is an accurate classifier that does not suffer from the same issues as those in a decision tree classifier. A key factor in the success of the random forest classifier is that it builds its trees randomly [1]. Decision tree classifiers, such as C4.5 and its predecessors used information gain, or another variation known as gain ration. This allowed the previously mentioned classifiers to target specific attributes instead of building multiple decision trees unlike the random forest classifier. Naturally, the

attribute selection algorithms used introduced some pitfalls such as bias and over fitting which gave the random forest classifier an advantage [2].

Iterating over every possible combination of features and building each combination as a decision tree takes massive processing power especially when there is a high number of features to select from [3].

Researchers have identified this issue and worked on multiple variations of random forest to reduce the number of decision trees to be grown. Some of the successful variations use Symmetrical Uncertainty which is an attribute evaluator and can be defined as the fraction between information gain and entropy. Gain Ratio, which measures the gain in information for classification in respect to the entropy of the features. And other attribute evaluation methods to select the feature combination that will yield the highest accuracy achieving trees and generate a random forest for these features rather than the entire dataset [4]. We will try to achieve a higher accuracy in predicting classes by using the binary particle swarm optimization algorithm along with the random forest classifier.

This paper aims to increase the prediction accuracy of the elected decision tree in a random forest model by using an optimization algorithm that selects the best feature to grow the tree. This is done by searching through the combinations of features and testing their accuracy. The best set of features is the set which yields the highest accuracy and the lowest number of features. The initialization of the initial swarm is done randomly, but throughout the successive iterations, the algorithm will start to converge. We will be applying our algorithm on the Wisconsin breast cancer dataset and the Titanic data set from the UCI machine learning repository. The metrics for evaluating the algorithm will be the classification accuracy, F-measure, recall and precision. Existing Solutions.

The rest of the study is organized as follows: Section 2 focuses on background information on data analysis and the most widely used classification algorithm. Literature review presented in Section 3. Section 4 and 5 providing the proposed approach and its implementation. Results in Section 6. Finally, the result and feature work in section 7.

2. BACKGROUND

Technology is an essential part of our lives, and we are becoming more dependent on our mobiles, PDAs, and other devices to help us with our day-to-day operations. These devices out-number the number of users by at least 2:1. We can gather the massive amounts of this data by a process known as digitization. In simple words, digitization is gathering raw, possibly unusable, data and into a format that can be used by a machine. This massive amount of raw data must be put into use [4].

This is where datafication comes in. Datafication organizes the raw information gathered through digitization to give us pointers as to what to expect from the generation of this data in the first place. For example, the data gathered from a random person's browsing history is digitized [5].

It is transformed into a format that can be handled by the machine. The datafication process will reveal crucial information pertaining to this person's work, environment, political interests, and other aspects of their life. This knowledge inferred from the datafication process came from the information gathered about the user's browsing history [1].

As stated earlier, the sheer amount of raw data generated by one person is massive and could be in terabytes. This dilemma opened research for new technologies to emerge to solve this problem. Albeit this is not the only problem to face, there is another problem, and it is the processing power required to handle complex and huge amounts of data. As is the case with most problems, a solution is inevitable to be found [6].

Part of the solution was provided by Google's Hadoop framework. This framework works on distributed machines to tackle the issue of the required processing power. These machines perform tasks or operations individually. Each task is a vital part of the result [7]. Hadoop has two main components. The first is the Hadoop File System (HDFS). The file system is responsible for allowing the master machine to gain access to the data provided by the slave machines that perform these individual tasks. The second component is the programming model MapReduce. Map phase breaks down a big problem into manageable chunks of smaller operations given to each slave machine.

The result is then reduced by removing redundant and irrelevant outputs. This technology is only one of many technologies designed specifically to tackle the issues of big data [7].

The second part of the problem is the need to store massive amounts of data on one or many smaller devices. As time passes, the basic file types such as images and sound records will only increase in size. This poses an open research problem. With these technologies, certain methods must arise. These methods must go beyond the ordinary statistical aggregation or any basic statistical function. These methods must be designed specifically to match the task at hand. In a nutshell, handling big data requires a person to master various aspects ranging from mathematics to computer science. The massive amount of data to be analyzed is conveniently named big data. Where does this data come from? Does it all come from one place, and where is it stored? Why is this data beneficial for scientists and analysts?

2.1. What Is Big Data and Data Mining

Big data impacts our lives daily without us even knowing it. Advertisements on Facebook or suggested videos on YouTube and many others employ big data to show content better suited to everyone based on the information the user has given. Big data once understood fully can benefit businesses, educational institutions, or even governments as the data generated can be used to predict future outcomes. From a business perspective, skilled businessmen are hired to analyze past and present data on a certain market to make an educated guess on what the market will be in a day or 10 years. The application of artificial intelligence in the business market will be the trend soon. As for now until we can fully utilize this data, it will be a future endeavor [8].

The problem with the processing of big data is not just because it comes in big chunks as stated earlier, but the fact that there are many sources and a wide array of formats to represent this data. Many other minor issues stare us in the face while examining this data such as, can this data be used securely? How to differentiate relevant from irrelevant data? This is the science of converting that information into knowledge through a process called data mining [7].

This process is built up of numerous levels but can be summed up in 3 levels or tiers [4]. The first tier is the big data mining platform. This tier describes the station a problem will be analyzed on. For small datasets a single node or machine is enough to yield the required results. On a larger scale this data has to be distributed among multiple nodes to increase the processing power allocated to each task within the problem as a whole.

The second tier is data semantics and application knowledge.[4]. This tier is narrowed down to the most elusive issues in this field. The first issue is data sharing and confidentiality. As speculated, some of the data can be very sensitive to certain individuals such as medical histories, places they have been, or even business transactions [9]. This issue essentially tackles the subject of who is authorized to use this information and how it should be used as to not directly affect the parties involved [7].

The second issue is the application domain of this data. As discussed earlier, this data comes in a variety of formats and from a variety of sources. What are the benefits one may hope to gain from sifting through all that data? Is it worth it from a business perspective? The third and last tier is techniques and algorithms.

There are numerous techniques to mitigate risks of incomplete data or biased results. The problems of data mining lie within the data itself. It could be incomplete thus it cannot convey the entire picture, or it could be biased based on the environment this data was collected from [4].

In this tier, risks like the ones mentioned above must be taken into consideration when designing and working on a data mining framework. The process of data mining is very systematic. To avoid the issues discussed earlier, the data must go through various stages to assure its integrity and conformity [3].

The goal of data mining is to find patterns that link this data together. To find these patterns, we must make sure that the data is clean of noise. Noisy data is any irrelevant information that could lead to the creation of false patterns. After we have eliminated the noisy data, we need to retrieve the relevant data from the various sources and their various representations. Conformity is a key aspect in the process of data mining [4].

The next step is data transformation. We must transform the data from their original formats into a format more suited for our goal. Once the transformation is complete, the process of data mining begins. This is the step where the unique and clever algorithms are employed [5]. In this stage a plain technique or a hybrid technique is applied to find a pattern within the data. When working with big data, it is inevitable to find multiple patterns. Once numerous patterns are found, these newly found patterns are evaluated based on measures, impact, and probability. During this step, the patterns found should be a part of the solution if the process was conducted correctly.

The last step is to represent this pattern or knowledge acquired in one of the various ways, such as induction rules or decision trees [5].At each step, the data is stored in data warehouses. The most researched topics in the field of data mining are the algorithms employed

themselves. How are they different? Why are there many algorithms that essentially perform the same process? When do we use each algorithm?

2.2. Data Mining Algorithms and Frameworks

Data mining algorithms are split into two major categories, descriptive and predictive [5]. The purpose of data mining is to essentially find the hidden patterns in the data and describe it in a form of representation. Representing the data and highlighting the relationships between the attributes of the data is called the descriptive model [4]. Another purpose of data mining is to predict an event based on previous events that may have presented similar symptoms, such as a person's expectancy to develop cancer at a later stage in life. Predicting an outcome for future events is the predictive model.

The Figure. below highlights the difference between the algorithms applied in each model.

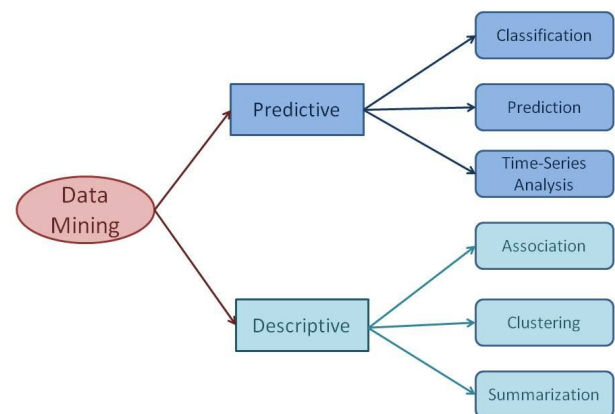


Figure 1: Data mining Models algorithms

As it is shown in the Figure. above, each model has its own set of techniques and algorithms. This did not prevent the breeding of these techniques to come upwith new hybrid techniques that have proven more efficient than either on its own. The prominent field in the predictive model is classification [10].

Classification is a form of categorizing the data based on a weight factor. Classification employs decision trees and neural networks to create a supervised learning environment to determine the weight factor for a set of data that can be used to categorize the attributes. This method is commonly used in modeling business or credit analysis [10].

On the other hand, clustering is the most prominent field in the descriptive model. Clustering is like classification in a sense it categorizes data into categories, but different on how it achieves this task. Clustering is a non-supervised training framework meaning it does not have a training set to help it identify the clusters the data will fall in to. As opposed

to classification, clustering does not categorize the data based on a weight factor but categorizes them according to their characteristics [5].

Clustering can be described as the process of identifying objects of the same class from an object-oriented approach. The attributes exhibited in each object determine the category a piece of data falls in. In this research we will be looking at classification algorithms in more depth rather than clustering algorithms.

2.3. Classification

Chronologically speaking ID3, Isometric Dichotomies, is one of the earliest classification algorithms [11]. ID3 is the steppingstone for the famous C4.5 and C5.0 classification algorithms.

The mechanism of this algorithm is simple. ID3 takes a training dataset that includes labels and attributes correctly classified. Then it builds a decision tree starting off with the attribute that yields the highest information gain, least entropy, least variance. Information Gain or simply referred to as IG is calculated based on how much information we can gain if we split the tree based on a given attribute [6]. The formula for information gain is represented as follows.

$$IG(T,a) = H(T) - H(T|a) \quad (1)$$

The formula can be simplified to become:

$$IG = \text{entropy}(\text{parent}) - [\text{weighted Average}] \text{entropy}(\text{child}).$$

The goal of this formula is to produce as many leaf nodes as possible. ID3 iterates over every attribute and calculates the information gain after each split. This algorithm was later replaced by the widely used C4.5 [7].

C4.5 introduced the management of continuous and discrete properties which allowed C4.5 to be applicable in more domains than its predecessor the ID3. The C4.5 algorithm works similarly to ID3 in the sense that it requires a training dataset to build a decision tree [10].

The criterion for selecting the attribute on which upon the data is split is chosen using a different way. Both algorithms rely heavily on information gain, but C4.5 goes beyond the information gain to calculate a gain ratio. What this extra step does is that it prevents bias for data with multiple entries [6]. This research will be focusing on a variation of the decision trees called “Random Forest”.

This variation enhances the prediction accuracy of the standard decision tree by building multiple decision trees. Random forest generates multiple decision trees because the standard decision trees suffer from over

fitting. Over fitting is when the decision tree is detailed to fit the training set that it loses its accuracy when applied to a real set. A big disadvantage of using random forests is the process in which it generates these trees. As its name suggests, it randomly selects the features to build the decision tree on without using any of the previously mentioned formulas [7].

This raises an issue of how long it will take to generate the forest if used on a set with a huge number of records and how much resources would it require. The random forest classifier is widely used in the classification model. It provides more accurate results than a decision tree [11].

Many scientists have tackled the problems of random forest and achieved promising results. As stated earlier, the random forest classifier generates multiple trees where each node is represents an attribute selected at random and this opened the field for the question; can this process be optimized to only select the important attributes? The difficulty of the task does not lie in reducing the number of the attributes, but how can these attributes be selected without compromising the accuracy? When the word optimization is uttered, optimization algorithms naturally spring to mind [10].

Optimization algorithms were designed for a specific reason, and it is to increase or decrease a certain aspect of any process. There are many optimization algorithms that can be applied in this problem and data scientists have been using the optimization algorithms arsenal to optimize the attributes in a way to preserve, and hopefully increase the classification accuracy in the random forest classifier [11].

This research uses a variation of the powerful optimization algorithm particle swarm optimization known as binary particle swarm optimization. Random forest classifier, particle swarm optimization, and binary particle optimization will be further discussed in later sections.

2.4. Particle Swarm Optimization

Any optimization algorithm’s main goal is to find the optimal set of solutions for a problem within the solution space. This is done usually by minimizing the problem’s cost function. “The particle swarm optimization algorithm is a global stochastic optimization algorithm based on the swarming behavior of animals in nature” [10].

The algorithm creates particles that act as agents. Each agent is responsible to search the solution. Each agent’s velocity is guided by its own individual best solution(lbest) as well as the swarm’s global best solution(gbest) [4]. As most optimization algorithms, PSO has a predefined number of iterations. The search stops when the objective is met, i.e., optimum solution

has been discovered, or the number of predefined iterations has been exhausted.

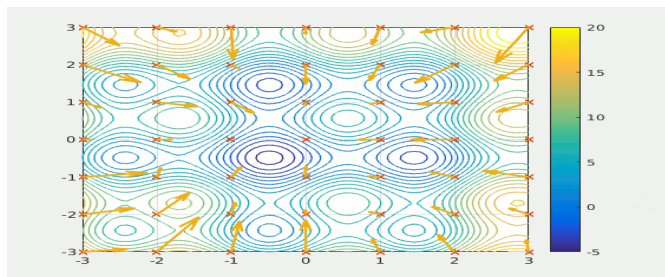


Figure2. Particles in the search space

2.5. Binary Particle Swarm Optimization

The above algorithm works for continuous problems where the parameters are real numbers [7]. This is not the case in this research. In this research, we employed the concept of Binary particle Swarm which instead of assigning the particles values from the real numbers set, they are assigned a string that consists of bits, 0 or 1. The velocity of the particle can be defined as the probability of X_{ij} . The algorithm constrains the velocity by using the sigmoid function. [9]

$$S(v_{ij}) = 1 / (1 + e^{-v_{ij}}) \quad (2)$$

$$X_{ij} = \begin{cases} 1 & \text{if } \text{rand}() \leq S(v_{ij}) \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

The value of $\text{rand}()$ is drawn from $U(0,1)$ and the function $S(v)$ is the sigmoid limiting transformation.

2.6. Random Forest Classifier

The random forest classifier is a supervised learning algorithm. As the name suggests, a random forest creates multiple “random” decision trees [12]. As stated earlier, a decision tree is prone to overfitting. Overfitting a decision tree means that a decision tree built using a certain dataset may not work as well with another.

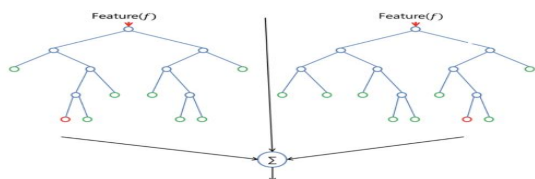


Figure3. Sample forest

The Figure. above illustrates how a random forest looks like with only 2 trees. A random forest builds each decision tree based on the best feature from a random subset of features. This results in a diversity that leads to a better model. Each feature in a random forest has an attribute called feature importance. Feature importance measures how much impact the presence or the absence of a feature will have on the prediction. [6].

2.7. Conclusion

It is obvious that data analysis plays an important role in our lives, sometimes without our knowledge. Storing, analyzing, and making sense of the information that we transmit online is a task with such magnitude that it requires special skills and new innovative ways of dealing with this information.

A technology could become obsolete over night as the market is ever changing. In data analysis, time and accuracy are of the utmost importance. The value of the technology is determined by how fast it is and how accurate its results are. Until this day, there is no perfect algorithm that can be applied to any problem. Each algorithm has its own pitfalls, but some are more efficient than others.

Hybrid algorithms are common as one algorithm solves an issue that may be present in another and vice versa. In the classification model, random forest classifiers are more accurate than decision trees as they provide less biased models. Even the random forest classifier has its own pitfalls which can be traced to attribute selection. Data scientists have addressed these pitfalls by applying optimization algorithms to select the features that best represent the dataset.

3. RELATED WORK

The field of machine learning and predicting future outcomes has been the focus for many researchers and research papers. One of the first works on machine learning and predicting future outcomes:

Archana et al [13].The authors improved approach consisted of a random forest for classification, Correlation Feature Selection for reducing noise, and suggested multiple algorithms for feature selection. The approach achieved a higher accuracy than the traditional random forest. Hassan [14], introduced a comparison between the genetic algorithm and the particle swarm optimization algorithm was conducted in this research to determine which optimization algorithm will find a better solution with less function calls.

Karegowda et al [15]. The authors analyzed the effects of using genetic algorithm as a search method with as a subset evaluation technique. The resultant feature subset is then tested with two supervised learning techniques, Back propagation neural network, and radial basis function network. The outcome of using the genetic algorithm with correlation feature selection showed improvement in both learning techniques. We used this research to study the extent an optimization algorithm could aid in the process of feature selection.

Chi et al [16]. The authors used an adaptive particle swarm optimization to determine the most suitable splitting variable when growing a decision tree. The approach starts by generating a standard CART decision tree using the gini impurity or gain ratio to determine the splitting variable. The generated tree is then optimized by using the adaptive PSO.

This approach increased the classification accuracy compared with CART and as the number of observations increased, the proposed algorithm outperformed the CART.

Fan [17] improved a new data classification method based on chaotic particle swarm optimization and least square-support vector machine. A new novel method based on the particle swarm optimization algorithm with least square support vector machine for data classification. The method is based on the chaotic optimization particle swarm and support vector machines. The COA (chaotic optimization algorithm) will start by processing the initial positions of the particles for the PSO which will lead to the replication of a chaotic PSO or CPSO. The resultant algorithm will optimize the parameters of the LS-SVM.

Chinnaswamy and Arunkumar[18], [19], they proposed a hybrid feature selection approach using correlation coefficient and particle swarm optimization along with extreme learning machine classifier, a feed forward neural network.

Hossam et al [20] proposed approach uses PSO feature selection wrapper to reduce the number of features to a manageable size that could represent dataset. With the reduced number of features, the random forest algorithm would grow the trees based on the results obtained from using the optimization algorithm. The technique used in this paper showed better performance than the traditional mail spam filters.

4. PROPOSED APPROACH

This section will go through the process of using the Binary Particle Swarm Optimization algorithm alongside the Random Forest classifier. The basic concept of the approach is to assign each particle an array of bits (0,1) to denote whether the feature exists or not. The assigning process is done randomly. The optimization algorithm starts and records the accuracy obtained during each iteration with the assigned parameters. The particles will move through the search space and change their bits until the maximum number of iterations has been exhausted. The output will be the global best accuracy. In this paper, a python tool was implemented to find the features to that record the best feature set to score a high accuracy[21], [22].

4.1. Proposed Approach

The proposed approach will use the random forest classifier, and an optimization algorithm to be used on the features. The goal of this approach is to improve the classification accuracy standard random forest classifier. A flow chart highlighting the processes in the proposed approach is shown below.

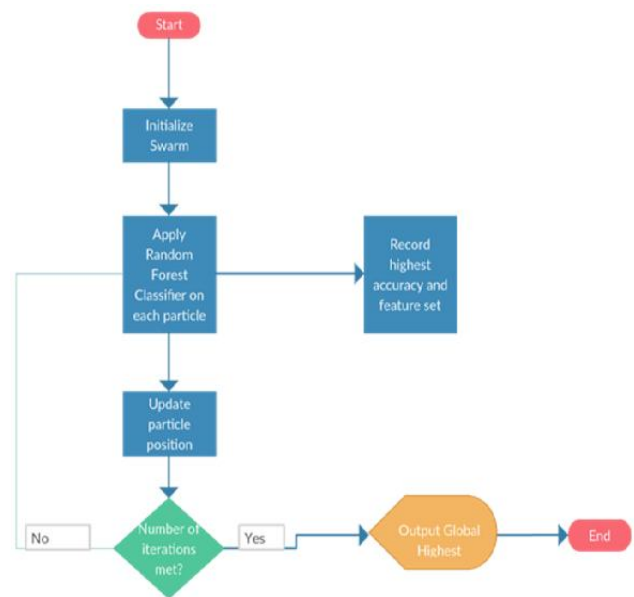


Figure4. Proposed Solution Diagram

The above flow chart can be further explained with the following pseudo code.

Input: $Ds\ Train = \{x_1, x_2, \dots, x_n\}$. The training set that contains the training examples and their respective classes.

Output: Prediction Accuracy

Method:

- Step 1:** Initialize the swarm population by assigning each particle a random array of bits denoting whether the feature exists in the solution or not.
- Step 2:** Pass the population swarm to the random forest classifier to acquire the accuracy of the proposed set of features.
- Step 3:** Record the accuracy and update the location of the particles. (Local Best)
- Step 4:** Repeat steps 2 and 3 until all iterations have been exhausted.
- Step 5:** Pass the global best to the random forest classifier for the last time to obtain the final accuracy based on the optimal set of features extracted previously.

The approach assumes that the dataset has already been prepared. The BPSO algorithm starts the optimization process by selecting a group of features and applying evaluation metrics to find the accuracy, precision, and recall given by each data set. For each iteration where the prediction accuracy increases the algorithm will continue and the most accurate set of features is saved as the local optimal solution. The optimization process will stop when the PSO algorithm has already exhausted its allocated number of iterations, currently set at 100 iterations. 100 iterations have been proven to be enough for the datasets used in

this research but can be increased to equal the number of combinations of the attributes in the dataset [23].

Once the optimization process stops, the process of growing the trees starts by using the features selected by the optimization algorithm. The whole process is further explained in the following section.

4.2. Development Environment and Libraries

The implementation of the proposed approach has been developed using Python (Pycharm Environment). The libraries used are as follows.

- Numpy: Package for scientific computing. Contains N-dimensional array objects and various functions to manipulate the objects within the array.
- Scikit-learn: Fundamental package for datamining and data analysis. Used for the implementation of the Random Forest Classifier and evaluation metrics.
- PySwarms: A Binary particle swarm optimization tool created by Lj Miranda [24].

4.3. Population Initialization

The first step is to initialize the population of the swarms. The class “BinaryPSO” will initialize the swarm population. The number of particles to be initialized must be passed as a parameter. The second parameter is how many dimensions there are in the dataset. The dimension parameter denotes how many features there are in the dataset. The final option in BinaryPSO is the options parameter. The options parameter is an array with the values for c1, c2, w, k, and p, which are learning factors, inertia, and the distance metric.

4.4. Calculating the objective function for each particle

This research uses an objective function from another paper [9], with a small alteration

$$f(X) = \alpha(1 - \text{scores}) + (1 - \alpha)\left(1 - \frac{n_f}{n_t}\right) \quad (4)$$

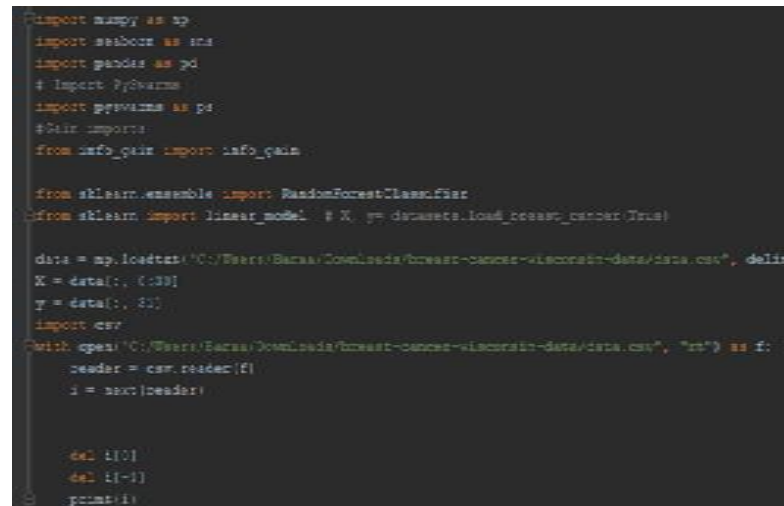
The alpha parameter is the decider of the tradeoff between scores (classifier performance) and the size of the features set n_f in respect to the total number of features which is denoted by n_t . The performance classifier can be the accuracy, precision, or others.

5. IMPLEMENTATION

The following section will explain in detail the steps and the code running the optimizer.

5.1 Importing Dataset

The first phase is to import the libraries and the dataset we will be using throughout the entire process.



```

import numpy as np
import seaborn as sns
import pandas as pd
# import PySwarms
import pyswarms as ps
#Data imports
from info_gain import info_gain

from sklearn.ensemble import RandomForestClassifier
from sklearn import linear_model # X, y = datasets.load_breast_cancer(True)

data = np.loadtxt("C:/Users/Bacaa/Downloads/breast-cancer-wisconsin-data/data.csv", delimiter=',', dtype=float)
X = data[:, 0:30]
y = data[:, 31]
import csv

with open("C:/Users/Bacaa/Downloads/breast-cancer-wisconsin-data/data.csv", "r") as f:
    reader = csv.reader(f)
    i = next(reader)

    del i[0]
    del i[-1]

    print(i)

```

Figure5. Importing Dataset

The libraries we will be using are NumPy to manipulate arrays, pandas for reading the dataset file which we will be using, and finally the sklearn library [25]. Once the data is uploaded, a variable (X) will save the indices of the attributes which we will be using to populate the particles. The variable (y) is the attribute we will be predicting. In Figure. 5 above, the dataset to be imported is the Wisconsin breast cancer dataset. It can be downloaded from Kaggle and many other places it has been a benchmark for many researchers [18].

The dataset is then split into test and training sets. The training set makes up 80% of the uploaded file, and the test set is the remaining 20%. This split ensures that most of the cases have been covered in the training set and the machine will not have to blindly guess an outcome it has not learned.

5.2. Particles

A binary particle swarm consists of a group of particles searching the solution space for the optimal solution to a given problem. In our case the optimal solution is an array with the least number of attributes with the highest prediction accuracy. Each particle in the swarm will be assigned to a random binary array, X_Subset. This array will hold information regarding which columns are being used in the training process and which columns are not. Each iteration of the program has multiple particles working simultaneously to find which particle will result in the highest accuracy. The array with the highest accuracy is saved as the local optimal solution.

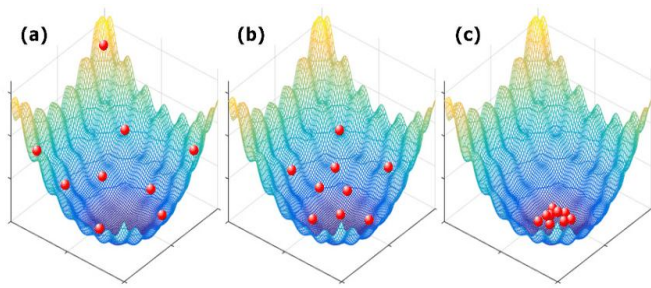


Figure6. Particles Converging

```
total_features = 30
# Get the subset of the features from the binary mask
if np.count_nonzero(m) == 0:
    X_subset = X_train
    X_subset_test = X_test
else:
    X_subset = X_train[:, m == 1]
    X_subset_test = X_test[:, m == 1]
# Perform classification and store performance in P
# classifier.fit(X_subset, y)
rfClassifier.fit(X_subset, y_train)
# print(y)
P = (rfClassifier.predict(X_subset_test) == y_test).mean()
scores = cross_val_score(rfClassifier, X_subset_test, y_test, scoring='accuracy', cv=10).mean()
# Compute for the objective function
j = (alpha * (1.0 - scores)
      + (1.0 - alpha) * (1 - (X_subset.shape[1] / total_features)))
# with open("iters.txt", "a") as myfile:
# myfile.write(str(m)+" accuracy of: "+str(P)+"\t"+str(j)+"\n")
return j
```

Figure 7. Particle training

The variable Total features will always be set as the number of attributes present in the dataset and disregarding the target attribute.

Figure.7 shows the processing done for each particle. The variable m is called a mask. The mask is responsible for converting each attribute into a binary array. Each iteration will train the subset against the training set then test it against the testing set. The scores variable will hold the accuracy of the model and enters it into the cost function. This process is done until the number of iterations has been met.

5.3 Calculating scores for best model

Once all the iterations have been exhausted, the global optimal solution is then trained again and tested for final evaluation. The code for the final test is shown in Figure.8.

```
opt_pos = optimizer.optimize(f, iters=10, fast=False)
# Create two instances of LogisticRegression
# classifier = linear_model.LogisticRegression()
rfClassifier = RandomForestClassifier(n_estimators=50)
a = pos
X_selected_features = X_train[:, pos == 1] # subset
X_selected_features_test = X_test[:, pos == 1]
scores = cross_val_score(rfClassifier, X_selected_features, y_train, scoring='accuracy', cv=10).mean()
# Perform classification and store performance in P
P = rfClassifier.fit(X_selected_features, y_train)
print(P.feature_importances_)
predicted = rfClassifier.predict(X_selected_features_test)
# Compute performance
subset_performance = (rfClassifier.predict(X_selected_features_test) == y_test).mean()
acc_test = metrics.accuracy_score(y_test, predicted)
# print (pos)
```

Figure8. Final Test

As Figure.8 illustrates, the optimal found solution is saved in a variable called pos. By this point, this variable only holds the masked binary array. The final solution is then converted into an array of the column names that we can use with the position checker program.

6. RESULTS

6.1. Evaluation Metrics

The performance of the proposed approach was evaluated using the classification accuracy, Recall, and F-1 Score [26].

Recall: Can be defined as the correctly identified actual positives or “How complete the results are”. It can be represented with the following Formula.

$$\frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{5}$$

When we apply the recall equation on the “Winsconsin Breast Cancer” dataset, we get 45 true positives and 2 false negatives. Simplifying the terminology, a true positive is when the machine correctly predicts a result in the test dataset. In other words, the machine has correctly identified 45 cases as patients with malignant breast cancer. A false negative is when the machine makes the wrong prediction that the result does not fall into the specified class, which in our example is a malignant tumor. Putting it all together the equation above is equivalent to:

$$\frac{45}{45+2} \tag{6}$$

The result is then multiplied by 100 to obtain the actual percentage that corresponds to the algorithm’s overall recall.

```
True Negatives: 65
False Positives: 2
False Negatives: 2
True Positives: 45
[[65 2]]
 [ 2 45]]

precision    recall    f1-score

0.0          0.97      0.97      0.97
1.0          0.96      0.96      0.96
```

Figure9. Breast Cancer Evaluation

As we can clearly see that the recall obtained by the algorithm is 96%. Doing the math results in 95.7 %, but the results are rounded to the closest integer.

Precision: Is defined as the correct positive identifications or “How useful the results are”. It can be represented with the following Formula

$$\frac{\text{True Positives}}{\text{True Positives+False Positives}} \quad (7)$$

The precision evaluation metric describes the percentage of the results that have been correctly classified. Based on Figure. 5 above, we can see that the count of true positives, or correctly identified positive classes, is 45. The count of false positives incorrectly identified positive classes is 3. Applying the same principal from equation 8 we end up with:

$$\frac{45}{45+2} \quad (8)$$

Solving equation 9 and multiplying the output by 100, to obtain the percentage, yields 95.74% precision. The result is rounded to the closest integer.

The results obtained and shown in Figure.8 were automatically generated by the Sickit Learn library also known as (SKLearn). The generated report in Figure.8 has 2 different rows for each evaluation metric. One is preceded by 0.0 and the other is preceded by 1.0. The 0 denotes false and the 1 denotes true. The true row will show the scores when using true positives and the false row shows the scores when using true negatives. For example, the precision column has a 97% when using true negatives and 96% when using true positives. The 97% can be obtained by replacing the true positive in the precision equation with true negatives.

$$\frac{65}{65+2} \quad (9)$$

F-1 Score: Is a measure of accuracy. F-1 score considers precision and recall and can be defined with the following Formula:

$$2 * \frac{\text{Precision*Recall}}{\text{Precision+Recall}} \quad (10)$$

The F-1 score, or F-measure is a harmonic mean between precision and recall.The above formula can be simplified to yield

$$F1 = \frac{2TP}{2TP+FP+FN} \quad (11)$$

By substituting the variables TP, FP,FN, true positives, false positives, and false negatives respectively with the values we have in Figure.9 we end up with the following equation.

$$F1 = \frac{2(45)}{2(45)+2+2} \quad (12)$$

The final result is 95.744%. The fact that the results are identical for each evaluation metric is purely coincidental for one of our test runs.

```
True Negatives: 64
False Positives: 3
False Negatives: 2
True Positives: 45
[[64 3]]
 [ 2 45]]

precision    recall    f1-score

0.0          0.97      0.96      0.96
1.0          0.94      0.96      0.95
```

Figure10.Test Run Sample 2

6.2. Application of the Proposed Approach on Winsconsin Breast Cancer Dataset and the Titanic Dataset

To measure how well the proposed approach compares to the standard random forest classifier, we applied the proposed approach on the Winsconsin Breast Cancer Dataset. The Winsconsin Breast Cancer Dataset has 30 features. Only 29 features were used in the test. The ID column was discarded. The 31st column was the diagnosis column and it only had 2 classes represented using a bit (0,1) denoting whether the patient has benign or malignant tumor.

The Figure.11 shows the results gained by both, a standard Random Forest Classifier, and the Random Forest with BPSO Feature selection classifier. The Figure. below compares our results with another research conducted in 2017 to evaluate the accuracy of the Random Forest classifier on the Wisconsin breast cancer dataset.

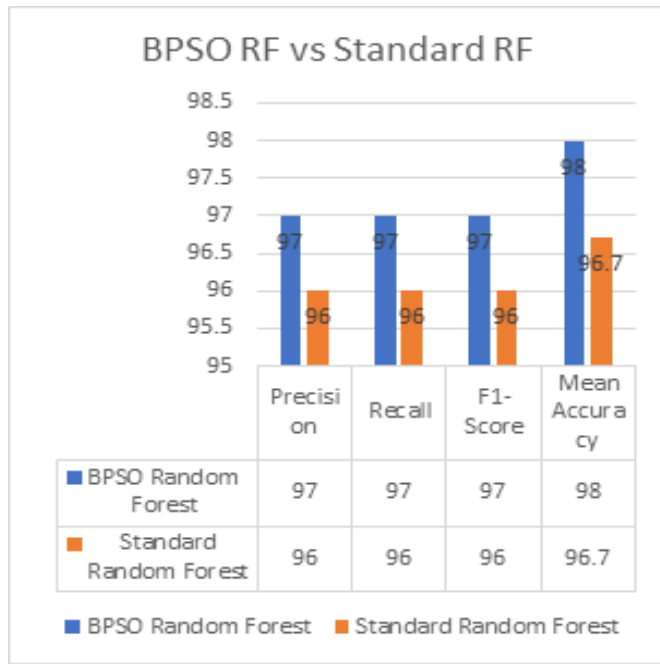


Figure7. Results of Comparison Breast Cancer

The Figure. above shows an increase in all the metrics after applying the binary particle swarm optimization algorithm on the random forest classifier. The results concluded that the optimization algorithm has improved the Random Forest classification, even if it was only by a small margin, for this dataset.

The other dataset we used to evaluate our work was the Titanic dataset provided by the UCI machine learning repository. The dataset consists of 8 attributes. We only used 7 of the 8 attributes as the passenger id does not play any role in determining whether the passenger has survived or not. The Figure. below describes the results obtained in previous research.^[18]. As we can see, there are many classification algorithms used in this research, but we will be focusing on the results obtained by the random forest classifier.

Table1. Algorithms' Results

Algorithm	Accuracy	F-Measure	Kaggle
Voting (GB, ANN, KNN)	0.86	0.82	0.794
Gradient Boosting	0.869	0.815	0.789
Calibrated (GB)	0.866	0.81	0.813
Random Forest	0.848	0.781	0.789

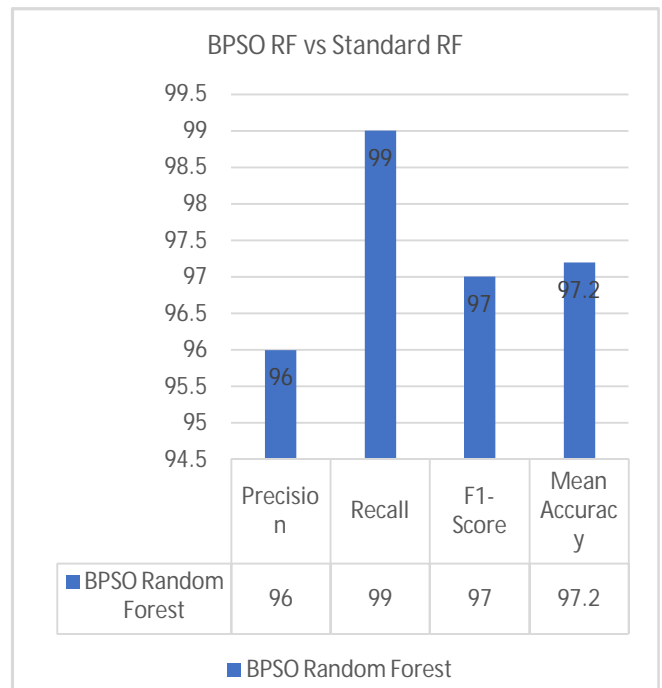


Figure8. Results Comparison, Titanic

7. VALIDATION

The main premise of this research is to reduce the number of “noise” attributes. To do so an optimization algorithm has been used to select a random set of attributes, pass it to the random forest classifier. After the initialization phase, depending on the cost function, the algorithm will keep on searching for the most appropriate attributes that describe the dataset. To keep track of the proposed algorithm and its results, we have used the “Play Golf” dataset. The dataset consists of 4 attributes, outlook, temperature, humidity, and windy. The goal is to determine whether a person will play golf given the weather conditions provided by the attributes.

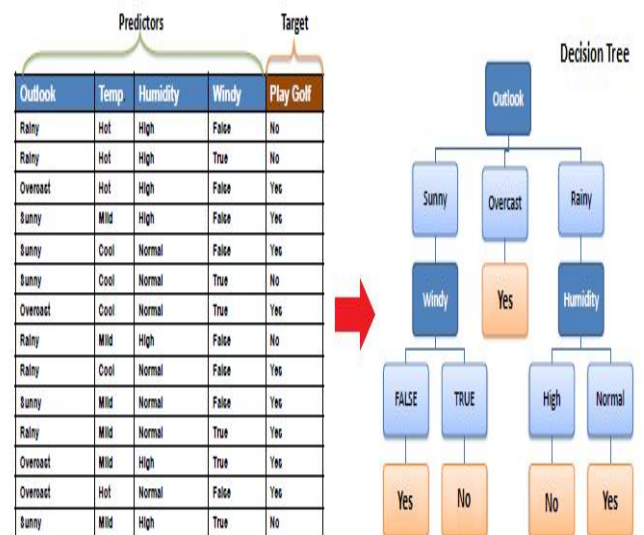


Figure9. Play Golf Dataset analysis

The Figure. above shows the attributes and the decision tree constructed. A typical decision tree is built by identifying the attribute with the highest information gain and used as a leaf node. In this example, the outlook attribute yields the highest information gain. By using the information gain formula, we can calculate the information gain for each split.

Table 2. Splitting The Dataset on EachAttribute

Outlook	Temperature	Humidity	Windy
0.247	0.029	0.152	0.048

The table above shows the information we gain by splitting the dataset on each attribute. The highest information can be obtained when splitting the dataset on the outlook feature for the first iteration. The outlook attribute is chosen as the decision for the first iteration. Repeat the same process for each iteration, calculate the information gain, then split the dataset furthermore on the attribute with the highest information gain. In 100 iterations, the enhanced random forest classifier achieved higher accuracy when the outlook attribute was used in the selected attributes set.

The data present in the play golf dataset is too small to work with. We will be demonstrating an entire test run and we will also be looking at the important pieces of code and the code we have used to validate our findings.

```

X_subset = X_train[:, n = 1]
X_subset_test = X_test[:, n = 1]
+ Perform classification and score performance in 2
+ classifier.fit(X_subset, y)
rfClassifier.fit(X_subset, y_train)
+ print(y)
y = (rfClassifier.predict(X_subset_test) == y_test).mean()
accuracy = cross_val_score(rfClassifier, X_subset_test, y_test, scoring='accuracy', cv=10).mean()
+ Compute for the objective function
f = (alpha * (1.0 - scores)
      - (1.0 - alpha) * (1 - (X_subset.shape[0] / total_features)))

+ with open('log.txt', 'a') as myFile:
+ myFile.write(str(f) + " accuracy of: " + str(100 * accuracy) + "%\n")
return f
    
```

Figure. 14. Particle Processing

Figure. 14 shows the processing aspect for each particle. For each particle the data is trained then testing by using X_Subset and X_Subset_test. For each particle, the subset of features selected for training and testing is different. For example, in Figure. 11, one particle could be training a feature

subset of only the “perimeter mean” and another particle could be training with another feature subset. In each iteration, each particle will test its subset and evaluate its accuracy with a cross validation of 10. Cross validation is a popular method for testing the accuracy of the model. What cross validation does is that it splits the testing set into 10 different sets and the goal is to measure how well the model performs in terms of accurate classification.

This validation method was used to try to reduce over-fitting as much as possible. As previously stated, over-fitted models tend to do well on the testing dataset but fail to achieve good results on another dataset of the same shape and premise. This is considered a concern because we did not want to run into the same pitfalls of the traditional decision tree. The variable j is our cost function. As stated earlier, the aim of this research is to achieve the highest accuracy by only using the attributes that have the highest weight without exposing the model to over-fitting. The optimization’s final result is displayed in Figure. 15.

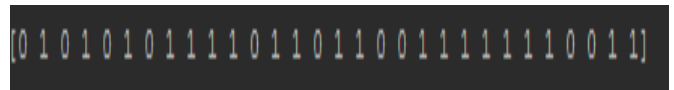


Figure 15. Optimization Result

The array is a binary representation of the attributes set that resulted in the highest accuracy and the least number of elements. A 0 denotes the absence of the attribute, and a 1 denotes the existence of the attribute. When multiple particles achieve the same accuracy with different number of attributes, the particle with the lowest number of attributes is selected. The resulting binary string also represents the locations of the attribute in reference to their positions in the original dataset file.

We use a separate project file to test the resulting attributes set to verify that the selected attributes set yields the accuracy our optimizer has achieved.

The input for the validation program is the binary string we obtained from the optimization process converted into its corresponding name in the original dataset as shown in the Figure. below.

```

data_mean = data[['perimeter mean', 'area mean', 'compactness mean', 'radius 99', 'texture 99', 'perimeter 99', 'smoothness 99', 'compactness 99', 'radius 99', 'texture 99', 'perimeter 99', 'smoothness 99', 'concave points mean', 'fractal dimension']]
predictors = data_mean.columns[0:10]
    
```

Figure 16. The input for the validation program

The number of predictors shown at the bottom of the Figure. allows us to select which attributes will be used in the prediction process. By default, the input for the

predictors is columns [0] the number of present columns we obtained in the optimization process.

The data mean variable only holds the attribute set we want to test and not all the attributes in the original dataset file. The data mean variable only holds the attribute set we want to test and not all the attributes in the original dataset file.

8. CONCLUSION & FUTURE WORK

Feature selection is a crucial part in creating good models for many reasons. One important reason is that it allows us to apply some form of cardinality reduction which results in reducing the number of attributes considered for building a model. Luckily, raw data almost always contains more information than we require to build a model. Naturally, dataset can include columns that provide little to no relevant information and analyzing these columns could be inefficient. There are many feature selection algorithms that are being used today such as filter methods, wrapper methods and others. This paper proposes a new approach to select features using the binary particle swarm optimization algorithm and apply the selected features with the random forest classifier to achieve the highest accuracy and the least number of features.

We have implemented a tool that accepts dataset in a csv file format as a parameter and uses the binary swarm algorithm to find the features that yield the highest accuracy. This is achieved by assigning each particle a random combination of features and applies the random forest classifier on those features. The highest accuracy obtained a particle is saved and the algorithm proceeds with the next iteration. In each iteration, if an accuracy higher than the saved is achieved, it will be recorded as the new highest. This process continues until the number of iterations has been met and the highest accuracy's features will be used for the final n-fold cross validation.

In the future we would like to examine the impact of larger datasets on the performance of the BPSO algorithm. The random forest classifier is known to suffer from performance issues when used with huge datasets. Decreasing the time required to build the model and select the features will be our next goal especially in larger and more complex datasets. We also would like to find the effects of using distributed processing on the quality of the results. Google's Hadoop will be our first choice, but we will expand our research to include more frameworks and see if other frameworks produce different results. Another goal we would like to achieve is to build our own distributed processing framework and compare it to the well-known frameworks and see where each framework outperforms the others and which framework is better suited for which tasks.

ACKNOWLEDGMENTS

This research was supported by Isra University, Amman, Jordan. We thank our colleagues who provided insight and expertise that greatly assisted the research.

REFERENCES

- [1] M. Greco, and M. Grimaldi. **What is big data? A consensual definition and a review of key research topics.***Proc. 4th International Conf. on Integrated Information*, Madrid,2017, pp. 33-39.
- [2] W. Dai, and A. Ji. **A MapReduce Implementation of C4.5 Decision Tree Algorithm.***International Journal of Database Theory and Application*, vol. 7, no.1, pp. 740-741, 2014.
- [3] T. Rosan, B. Al-Shargabi. **A New Maturity Model for the Implementation of Software Process Improvement in Web-Based Projects.***journal of Digital Information Management*. vol. 12, no.2, 2017.
- [4] S. Sulaiman and R. Salam,**Risk Analysis and Web Project Management.***Journal of Software*, Vol. 4, no. 6, August 2009.
- [5] F. Chen, P. Deng, and J. Wan. **Data Mining for the Internet of Things: Literature Review and Challenges.** *International Journal of Distributed Sensor Networks*, vol. 11, no. 8, August 2019.
- [6] F. Alzyoud, N. AL Sharman. **Smart Accident Management in Jordan using Cup Carbon Simulation.** *European Journal of Scientific Research*, vol. 152, no. 2, 2019.
- [7] H. Abualese, and B. Al-Shargabi. **A New Trust Framework for E-Government in Cloud of Things.***International Journal of Electronics and Telecommunications*,vol. 58, no.1, pp. 431-440, 2019.
- [8] K. Deepshri and Kamathm. **A Survey on Techniques of Data Mining and its Applications.***International Journal of Emerging Research in Management & Technology*, vol. 6, no.2, pp. 332-339, 2018.
- [9] S. Nikam. **A Comparative Study of Classification Techniques in Data Mining Algorithms.***Oriental Journal of Computer Science and Technology*, vol. 8, no.1, 2018.
- [10] O. Al-Haj Hassan, A. Abu Taleb and A.iMaaita.**An efficient and scalable ranking technique for mashups involving RSS data sources.***Journal of network and computer applications*. Vol.39, 2014.
- [11] Chaudhary, S. Kolhe and R. Kamal. **An improved Random Forest Classifier for multi-class classification.***Proc. 3th International Conf. on Computer Supported Cooperative Work in Design*, Malaysia,2019, pp.201-208.
- [12] B. Al-Shargabi and M. Hassan.**A Novel Approach for the Detection of Road Speed Bumps using Accelerometer Sensor.** vol. 9, no.2, 2020.

- [13] L. Zhou, H. Wang and W. Wang. **Parallel Implementation of Classification Algorithms Based on MapReduce.***Telekomnika*, vol.16, no.5, pp. 1087~1092, September 2018.
- [14] T. Rousan and S.Sulaimin. **Supporting Architectural Design Decisions through Risk Identification Architecture Pattern (RIAP) Model.***WSEAS transactions on information science and applications*, vol.4, no.6, pp. 611-620,2009.
- [15] R. Archana, B. Cohanin and O. Weck. **A Comparison of Particle Swarm Optimization and the Genetic Algorithm.** *IEEE Internet Computing*, vol 21, no.2, pp.86 - 90, 2017.
- [16] H. Abualese, P. Sumari and M.Rasmi.**Utility classes detection metrics for execution trace analysis.***Proc. 8th International Conference on Information Technology (ICIT)*. IEEE.2017.
- [17] T. Rousan. **An Investigation of User Privacy and Data Protection on User-Side Storage.** *International Journal of Online and Biomedical Engineering (iJOE)*, vol. 15 no. 9, pp. 17-30, 2019.
- [18] G. Karegowda, C. Ciufudeanand M. Jayaram. **Comparative Study of Attribute Selection Using Gain Ration and Correlation Based Feature Selection.***International Journal of Information Technology and Knowledge Management*, vol 11, no.2, pp.88 - 97, 2018.
- [19] C. Chi, Y. Cho and H. Lee. **Improving Tree-Based Classification Rules Using a Particle Swarm Optimization.** **Christos Emmanouilidis.***Proc. 19th International Conf. on Production Management Systems (APMS)*, Italy ,2020, pp. 121-129.
- [20] L. Fang. **A New Data Classification Method Based on Chaotic Particle Swarm Optimization and Least Square-Support Vector Machine.***IEEE Internet Computing*, vol 21, no.2, pp.86 - 90, 2018.
- [21] T. Rousan, S Aljawarneh.**Using cloud computing for E-government: A new dawn.***Advanced Research on Cloud Computing Design and Applications.2015*
- [22] Chinnaswamy and R. Srinivasan. **Hybrid Feature Selection using Correlation coefficient and Particle Swarm Optimization on Microarray Gene Expression Data.***IEEE Software*, vol 37, no. 1, pp.112 - 116, 2020.
- [23] H. Hossam, I. Aljarah and B. Alshboul. **A Hybrid Approach based on Particle Swarm Optimization and Random Forests for E-mail Spam Filtering.** **Lecture Notes in Artificial Intelligence.***Computer*, vol 52, no. 9, pp. 51-58, 2019.
- [24] S. Al Awaida and B. Sharabi. **Automated Arabic Essay Grading System Based On F-Score and Arabic WorldNet.***Jordanian Journal of Computers and Information Technology (JJCIT)*, vol. 5, no.3, pp. 170-180, 2019.
- [25] J. Kennedy. **Particle Swarm Optimization.****Risk Analysis and Web Project Management.***Journal of Software*, Vol. 4, no. 6, August 2009.
- [26] L. Mirand. **A Binary Particle Swarm Optimization Tool.**<https://github.com/ljvmiranda921>". accessed:2021.