DOI: http://dx.doi.org/10.26483/ijarcs.v8i9.5173

ISSN No. 0976-5697

Volume 8, No. 9,November-December 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

RAPID PSO BASED FEATURES SELECTION FOR CLASSIFICATION

Surendra Kumar
Research Scholar,
Dr.K.N.ModiUniversity,Newai
Rajasthan,India

Dr.Harsh Kumar
Associate Professor,
HimalayanGarwalUniversity
Pokhara,India

Abstract: Sentiment analysis is one of the most promising research areas in the field of data analysis. In sentiment analysis we do classification of the data samples into positive, negative or neutral classes and bring out proper conclusions from the data. From movie reviews to the twitter data and many others we can impose sentiment analysis. There are many classification models that can be used for sentiment analysis. Some of these models are effective in doing the analysis and some are not. Though the effectiveness and efficiency of the analysis depend on many factors one of the major factors is the underlying feature selection process of the classification models. The features are the attributes of the data samples. Instead of classify the data samples based on the complete set of attributes one can build a model which uses the subset of attribute. Our proposed framework integrates a metaheuristic optimization technique which is a variation of Particle Swarm Optimization which we have named as RAPID-PSO with the multiple classification models which extract a subset of features from the complete set of features. Based on this sub set of features the framework farther classifies the samples. The result shows that our proposed frame work outperforms the other frameworks that use other classical optimization techniques such as –Grid search, Gradient Descend, Classical-PSO, Multiple-PSO, IPSO in terms of effectiveness and efficiency.

Keywords: Metaheuristic Optimization, Machine Learning, Grid search, Gradient Descend, Sentiment Analysis, PSO.

I. INTRODUCTION

We are in an age of internet and it is become the part of our daily life as social media. Our daily life is affected by the people around us who may be near or for away from us. The opinion and ideas given by other people may change our own opinion. In this era sentiment analysis provides a new way to learn the opinions about large population in the world. To learn about the opinions of a large set of people it is required to collect correct samples of opinions from people around the world [1]. Current use of web 2.0 has increased many activities like blogging, weblogs, tagging, opinions, comments, suggestions and feedback on Facebook, twitter, LinkedIn[2]etc.

Swarm intelligence is the discipline which resembles the natural phenomena where individuals work collaboratively to accomplish certain task. In Swarm intelligence interaction with in a swarm does not happen under any supervision rather it happens due to primitive instincts of the individuals. The main mechanism of this interaction is called as stigmergy which uses agents or actions for indirect coordination [3]. According to [3] Stigmergy is a well-defined form of self-organization. It produces complex structure, seemingly intelligent structures, without need for any control, planning or any direct communication in between the individual agents. It supports efficient collaboration between swarm agents, those who lacksof any memory, intelligence, action or any type of individualinteraction of each other. The PSO algorithms is more interesting and attracted to the researcher in the field of fuzzy logic system, neural network, optimization, pattern recognition, robot technology, signal processing etc[4].

Feature selection methods can be used to find out and remove irrelevant unneeded, and duplicate attributes from a data set that do not used to find the accuracy of anyprediction algorithms or which may decrease the accuracy of the algorithm or model[5].Feature selection is a challenging task because of the large size of the search space. The search space size is increases exponentially with respect to the number of features available in the data set [5].

The main objectives of feature selection are to provide the faster and more cost-effective, to provide a better understanding of the defined process that generatethe data, and to improve the prediction performance of the predictor.

Feature selection serves as a tool for the pre-processing before a classification problem. In the process of feature selection we have to defined the feature first then the selection process of feature and after that we do extraction and reduction process. In feature selection process identification of feature include opinion word, frequency, part of speech etc. Feature selection is also known asattributes selection or variable selection. Normally feature selection algorithms are divided in three parts, one is filter approach, wrapper approach and embedded approach. The filter approach is generous and less expensive computationally while wrapper approach uses the many classification algorithms and evaluation procedure. Normally wrapper approach produces more accurate result [6].The embedded approach uses the combination of filter and wrapper approach in which searching of optimal feature is combine with the classifier construction and searching can be done on the whole space of hypothesis and feature subset.The advantage of this method is that it interact with the classification model[7].

Sentiment classifier is last phase of the sentiment analysis after preprocessing of the data (text data). The SC classifies the text as negative or positive. It is well defined for short text as well as large text in twitter, blogs, facebook status and any product review [8]. The classifiers are also used to find out the subjectivity of a text-document where we try to find out the positiveness or negatineness of text in the document. But sometimes it fails to find the accurate result because of different culture of humans, way of expression, use of slangs, etc[9].Classification is the most important in machine learning to predict the category and class of a value [10]. Each and every data is represented by its features that specify its data space representation [11]. The manner in which data is represented is a key factor which affects the classification performance [12].

PSO (Particle Swarm Optimization) is a metaheuristic technique which is used for optimization as well as to explore the search space of any problem to maximize or minimize parameters based on a particular objective function. There are various variants of PSO and lots of meta-heuristic algorithms are available to find the optimal solution of the engineering problems. Wang et.al[13] proposed an optimization method that uses differential evolution (DE) mutation operator to the accelerated particle swarm optimization (APSO) algorithm and find a new optimization method called as DE type mutation in to APSO and a hybrid algorithm is called differential evolution accelerated particle swarm optimization (DPSO) a new variant of PSO. Wu Deng et.al[14] use a standard hybrid swarm intelligence optimization algorithm called GA-PSO-ACO which combines the evolution technique of Genetic algorithm, ACO and PSO. They simulate the results and finds that GA-PSO-ACO algorithm improves the efficiency in TSP problem AfsanehMahanipour et.al[15] uses a variant of PSO called IPSO in which some periodic stages are involved in evolution and whole population is shuffled and particles are reassigned. After performing simulation IPSO find better ability to get optimum global best than other existing PSO algorithms.

Another method is found to the optimization the composite material Alonso Maria et.al [16] used PSO algorithm as a new variant in BOSS Quattro optimization toolbox. They compared evolutionary algorithms and PSO and used software BASS Quattro for the comparison. The PSO and genetic algorithm uses a random initial population and to evaluate the fitness value it uses a fitness value. The updating and search mechanism are done using random techniques.

II. ALGORITHM BACKGROUND

A. Particle Swarm Optimization

PSO was first proposed by James Kennedy and Russell C. Eberhart in 1995 [17], for their experiment they simulated the swarming behavior or social behavior of individuals like flock of birds, ant, bee, insect and fish schooling and came to the conclusion that this behavior can be used for the process of optimization. The individuals are also called as agents or particles in the search space. A swarm is group of same type of agents. They perform the elementary task and communicate locally among the group in their search space without any type of centralized control. Eberhart and Kennedy found that the word particle is more appropriate than point since particle is associated with velocity and acceleration.

The position of a particle i, in the search space is normally represented by a vector $\vec{x}_i$. Each particle of the swarm is also associated with particle velocity $\vec{v}_i$ which is needed to update in each iteration of the algorithm.The movement of the particles in the search space is governed by two vectors-personal based solution ($\vec{p}_{bst}$) and global best solution ($\vec{g}_{bst}$). $\vec{p}_{bst}$and$\vec{g}_{bst}$ are calculated by using an objective function $f(x)$. The position vector $\vec{x}_i$ is updated based on above two vectors $\vec{p}_{bst}$ and $\vec{g}_{bst}$ so the basic entities that we need to maintain for PSO algorithm are listed below[18][19].

- Position vector of the particle $\vec{x}_i$
- Velocity vector of the particle $\vec{v}_i$
- Particle personal based solution vector $\vec{p}_{bst}$
- Global based solution vector $\vec{g}_{bst}$
- An objective function$f(x)$.

The updation of position of the particles happens based on the following two basic equations

$$\vec{v}_i=\chi\,[\vec{v}_i + C_1\,\overrightarrow{\in_1}\,(\vec{g}_{bst} - \vec{x}_i) + C_2\,\overrightarrow{\in_2}\,(\vec{p}_{bst} - \vec{x}_i)] \qquad \textbf{(1)}$$

$$\vec{x}_i = \vec{x}_i + \vec{v}_i \qquad \textbf{(2)}$$

Where χ is called the constriction factor and χ < 1. This is basically acted like friction slowing the particles so that linear exploration is auctioned.

C_1, C_2 : control the relative attraction to the global best and personal best found solutions

$\in_1$,$\in_2$: Vector of random variables drawn with uniform probability from [0, 1].

```
FOR EACH particles i
Randomly initialize $v_i, x_i = p\_i$
    Evaluate $f(\vec{p}_{bst})$
    $\overrightarrow{g}_{bst}$= avg max{$f(\vec{p}_{bst})$}
REPEAT
    FOR EACH PARTICLE  $i$
    Update particle position $\vec{x}_i$ according to equation 1 and 2
    Evaluate $f(\vec{x}_i)$
        update personal best
    IF $f(\vec{x}_i) > f(\vec{p}_{bst})$ then
        $\vec{p}_{bst}$= $\vec{x}_i$
        //update global best
    IF$f(\vec{x}_i) > f(\vec{g}_{bst})$ then
        $\vec{g}_{bst}$ = avg max {$f(\vec{p}_{bst}$}
Until termination criteria reached
```

B. Rapid Particle Swarm Optimization

The Rapid particle swarm optimization is an enhanced version of PSO. It carries the similar concept of swarm movement to locate the optimum value. The process begins with the initialization of randomly chosen population of particles in the search space. In standard PSO, the particles update their position and velocity for the next move based on the particle's own best position and the global best.

During the search of the optimum we use tomaintain the diversity among the particles or solutions in the search space which can also be maintained by using some degree of randomness.

The major difference in Rapid when compared with PSO is that it updates the velocity as well as the position of the particles by using only, thus speeding up the search exploration. Using only accelerates the convergence of the algorithm. The updating rules in the Rapid PSO has been shown in the following –

$$v_i^t = v_i^{t-1} + \propto \in_n + \beta(g_{bst} - x_i^{t-1}) \qquad \textbf{(3)}$$

After updating the velocity vectors of the particles, the position vector of the particles can be updated as follows-

$$x_i^t = x_i^{t-1} + v_i^t \qquad \textbf{(4)}$$

By neglecting the velocity updation of the particles we can further increase the convergence of the swarm. We can write the update of the location in a single step.

$$x_i^t = (1-\beta)x_i^t + \beta g_{bst} + \alpha \in_n \qquad \textbf{(5)}$$

Further Modification can be done to Rapid PSO by clamping the region of initialization of particles and the velocity used if the updationis according to the equations (3) and (4).

III. PROPOSED MODEL

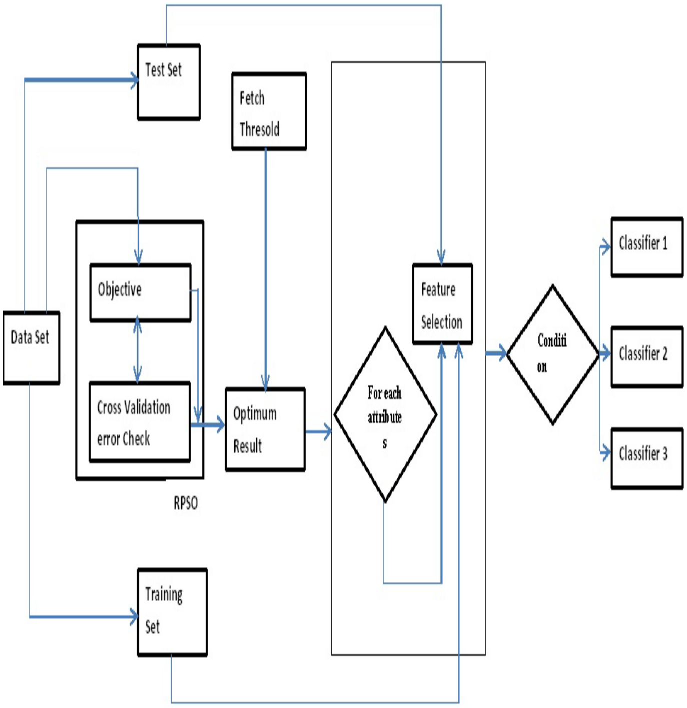


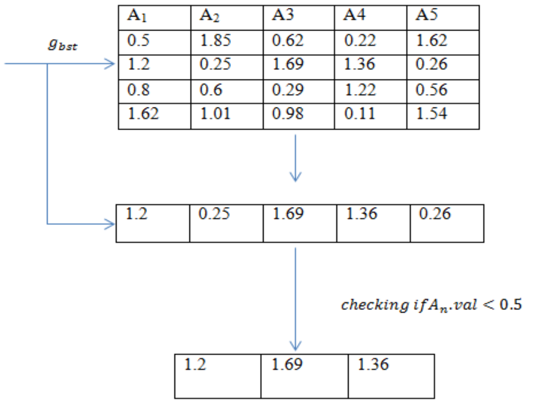
Fig: 3.1 Proposed model

```
Algorithm: RAPID-PSO
Input: RAPIDPSO parameters,
$R_{min}\,[A_1, A_2, \ldots A_n]$ , $R_{max}\,[A_1, A_2, \ldots A_n]$,
$F_{th}$ Output: Optimal Feature
Subset.
    // Initialization of the
particles
REPEAT
   FOR EACH particle $i$ in $S$
     Randomly initialize $\vec{v}_i$ ,
$\vec{x}_i = \overrightarrow{pos}_i$
          $\vec{g}_{best} = \arg\max\,\{f(\overrightarrow{pos}_i)\}$
        IF  $\vec{g}_{best}$ is better than
$\vec{g}_{current\_best}$ THEN
            $\vec{g}_{current\_best} = \vec{g}_{best}$
        END IF
    END FOR
       // Update particles position
       // Using Equation (3) , (4) and
(5)
  UNTIL Termination Condition reached
FOR EACH features $i$ in
$Feature\_set$
IF  $\vec{g}_{current\_best}[i]$ is less than $F_{th}$ THEN
                 DELETE $\vec{g}_{current\_best}[i]$
                 RETURN $\vec{g}_{current\_best}$ with
one element less
        END IF
   END FOR
```

Classical PSO works on the principle of position updation based on the velocity updation. A particle continuously changes its velocity based on the local history (p_{bst}) and global history (g_{bst}) and this changes on the velocity in turn changes the position of the particles as shown in (eq.1) and a search space needs to be define for the particles which is nothing but the variables values used in the objective function also called as score function. In every iteration to update the velocity and position we need to measure the score of the particles. In our model we have identify a search space which is dependent on the data set in hand and also instead of using classical PSO we have used a variation of PSO called Rapid PSO (RPSO).

In this model we have used Rapid to select a subset of features. For this our algorithm initializes random particles on the search space. The algorithm makes them iterate for a predefined number of times. After all the iteration the whole swarm (Collection of particles) converges to a particular state. In this state the algorithm select attribute or feature based on a threshold value. The particles in Rapid PSO are represented as a vector$[A_1 A_2 A_3 \ldots \ldots .. A_{n-1} A_n]$. Here each of the dimensions represents a feature of the data set. In the process of feature selection we get a solution vector $[A_1 A_2 A_3 \ldots \ldots .. A_{r-1} A_r]$ where$r < n$. This result vector has been extracted from the global best g_{bst} vector of the swarm at the final iteration of the Rapid PSO searching process. The extraction process is done by using a threshold measurement on the values of each dimension of the final global best g_{bst} vector.

Suppose we have a dataset of 5 features. The following is the configuration of the swarm is our algorithm at the final iteration.



A1	A2	A3	A4	A5
0.5	1.85	0.62	0.22	1.62
1.2	0.25	1.69	1.36	0.26
0.8	0.6	0.29	1.22	0.56
1.62	1.01	0.98	0.11	1.54

1.2	0.25	1.69	1.36	0.26

1.2	1.69	1.36

Fig 3.2 Feature Selection Criteria

A. Objective function

For the process of feature selection in our model we have used an objective function which is dependent on the attributes of our data set. The objective function can be expressed as below:

$$f(A_1, A_2, \dots . A_n) = A_1 + A_2 \dots \dots . . A_n$$

In our optimization process our main motive is to find the g_{bst} position for which the objective function gives a maximum value in every iteration of the process. We have decided the ranges of the attributes $(A_1, A_2, A_3, \dots \dots \dots . A_n)$ as the number of valid cells in the data set we are using. Due to the different length of ranges we get an asymmetric search space. In The particle vectors the variations of the attributes which are having a small range will have less variation and the attributes having large range we have more variation. Due to the large range attributes g_{bst} position will be having more probability of having them and at the last conversed position (last iteration) the particles will be having a g_{bst} position which will be mixture of all the attributes. The tendency moving towards a g_{bst} containing large range attributes will be more due to our objective function.

To decide the threshold using which we are selecting the attributes of the final g_{bst} position is based on the average value of the low ranged attributes.

Suppose we are taking ranges of attributes of a five attribute dataset as bellow:

$A_1 \to 0 \dots \dots 5$
$A_2 \to 0 \dots \dots \dots \dots \dots \dots \dots \dots . . 100$
$A_3 \to 0 \dots \dots \dots \dots .7$
$A_4 \to 0 \dots \dots \dots \dots \dots \dots 50$
$A_5 \to 0 \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots . . 200$

$$Thresholdvalue = \frac{\dfrac{Totalnumberofvalidcellsin}{3\ topmostlowrangeattributes}}{3}$$

In our example A_1, A_2, A_4 are the three top most low range attributes

$$Threshold = \frac{5 + 7 + 50}{3} = 20.66$$

B. Model Explanation:

Figure (3.1) represents a generalized model of our proposed work. It contains two basic modules, the first module is Rapid PSO module which selects the optimum attributes or feature vector based on the g_{bst} position of the final swarm. To selects these attributes or feature vector it uses an objective function as defined in section (3).

The result obtained from this module then goes through a filtration process which is based on a threshold value. We select the necessary attribute by putting conditions on the attributes which is dependent on that threshold.

Let

$$D(t): Datasetattimeinstancet$$
$$V_{th}: Thresoldvalue$$

$$\overrightarrow{V_{acc}}:\ FeaturevectorobtainedfromACCPSOmodule$$

The filtration process

```
FOR EACH FEATURE  $f_i$ in $\vec{V}_{acc}$
    IF $f_i$ $value < V_{th}$
        SELECT the-Feature  $f_i$
        END IF
    END FOR
```

Fig: 3.3 Filtration Process

The result obtain from this module which is a dataset with less number of features is then fetch to the second module which is an integration of various classifiers. Based on the characteristics of the dataset we decide the classifier to be used.

For example it the dataset of the nature “Text” that we use SVM classifier because SVM classifier best suited for text data. Not only text data but any bioinformatics data can also be well classified by SVM. For small data set we can use statistical model of classification such as Naïve Bayes classifier.

To decide the classifier to be used we take the parameters such as number of attribute in pre-process data set, number of rows in the data set, type of data in the dataset etc.

IV. IMPLEMENTATION

A. Tools Used

We have used python as the basic implementation tool for modeling our framework. Python is a high-level programming language (interpreted language) whose design philosophy emphasizes code readability.The power of python lies in the fact that the syntax of python allows programmers to express concepts in fewer lines of code which might use more lines in languages like C++ and Java. Few of the basic features of python are listed below –

- Python supports dynamic type system

- It has support for automatic memory management and multiple programming paradigms such as object oriented, imperative, functional programming and procedural styles.

- Due to the open-source nature, python can be ported to many other platforms.
- Python uses standard libraries. For scientific calculations python is a very effective and efficient programming language.

For our implementation we have used python version 2.7. Python 2.7 provides an environment where we can integrate packages and write codes. The packages used in implementation are described in the following table.

Table. 4.1 Used packages

Package	**Description**
Numpy	Large multi-dimensional arrays of arbitrary records can be processed efficiently using Numpy with python and integrates high-level mathematical functions to operate on these array and open source.
SciPy	It is open source software for mathematics, science and engineering experiments and depends on Numpy and provides a fast way to manipulate N-dimensional array.
Scikit-Learn	Scikit-Learn is a machine learning library for python programming language which is built upon Numpy and SciPy. It includes most of the basic algorithms for classification, regression and clustering like Support Vector Machines, random forest, gradient boosting, k-means.
Pandas	It is a python package for data analysis. It provides fast, flexible and expressive data structures for relational or labelled data processing.It is to give a real world insight for data analysis in python
Matplotlib	Matplotlib is a plotting tool for python programming language and NumPy. It is based on object-oriented API for integrating plots into applications using general-purpose GUI toolkit (wxPython, Qt, GTK+).

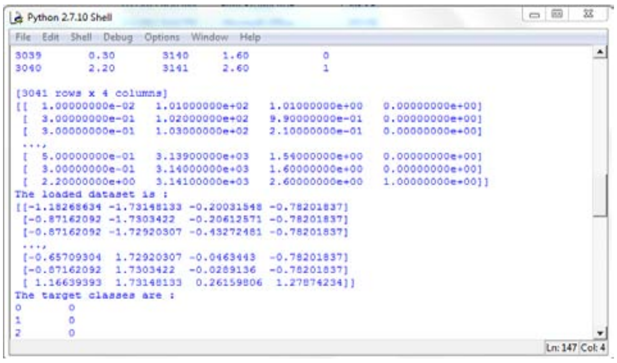
B. Dataset

The following is the python code snippet from our basic code which load the sentiment dataset –

```
defload_data_set():
from pandas import DataFrame, read_csv
importmatplotlib.pyplot as plt
import pandas as pd
import sys
df = pd.read_csv('senti_data.csv')
print(df)
numpyMatrix = df.as_matrix()
```

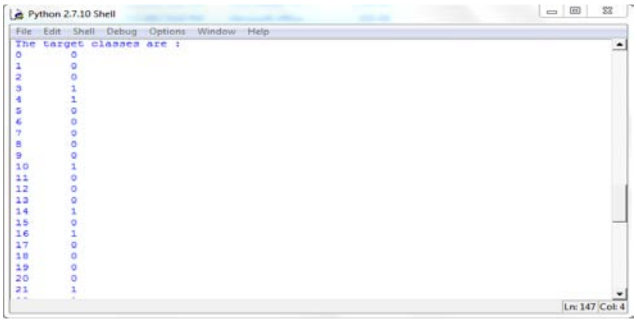
```
y = df["sentiments"]
X = preprocessing.scale(X)
return X, y
```

we have implemented a method ***load_data_set***()which loads the dataset. The output of the above code snippet is shown in the following.



```
3039        0.30        3140     1.60          0
3040        2.20        3141     2.60          1

[3041 rows x 4 columns]
[[  1.00000000e-02   1.01000000e+02   1.01000000e+00   0.00000000e+00]
 [  3.00000000e-01   1.02000000e+02   9.90000000e-01   0.00000000e+00]
 [  3.00000000e-01   1.03000000e+02   2.10000000e-01   0.00000000e+00]
 ...,
 [  5.00000000e-01   3.13900000e+03   1.54000000e+00   0.00000000e+00]
 [  3.00000000e-01   3.14000000e+03   1.60000000e+00   0.00000000e+00]
 [  2.20000000e+00   3.14100000e+03   2.60000000e+00   1.00000000e+00]]
The loaded dataset is :
[[-1.18268634 -1.73148133 -0.20031548 -0.78201837]
 [-0.87162092 -1.7303422  -0.20612571 -0.78201837]
 [-0.87162092 -1.72920307 -0.43272481 -0.78201837]
 ...,
 [-0.65709304  1.72920307 -0.0463443  -0.78201837]
 [-0.87162092  1.7303422  -0.0289136  -0.78201837]
 [ 1.16639393  1.73148133  0.26159806  1.27874234]]
The target classes are :
0        0
1        0
2        0
```

Figure 4.2(a). Dataset after conversion to matrix



```
The target classes are :
0        0
1        0
2        0
3        1
4        1
5        0
6        0
7        0
8        0
9        0
10       1
11       0
12       0
13       0
14       1
15       0
16       1
17       0
18       0
19       0
20       0
21       1
```

Figure 4.2(b). Loading Dataset

After loading the dataset we have imposed our classification model. We first train our model using the loading dataset and then validate the model using cross validation. Instead of training the model using all the features (attributes) of the dataset we use k– topmost features selected by Rapid PSO.

To do the feature selection we have implemented a method ***FEATURE_SELECTION***( )which returns dataset containing less number of features or attributes. This returned dataset is then used to train the model.

The***FEATURE_SELECTION***( )method used above is the method where the logic of Rapid PSO has been used. This method takes the original dataset in the variable $\boldsymbol{X}$(with the original number of attributes) and returns a dataset with less number of features$\boldsymbol{X_new}$.

$\boldsymbol{INTEGRATED_CLASSIFICATION}()$is the method implemented which takes the $\boldsymbol{X_new}$and $\boldsymbol{y}$for training the classification model. This method includes various classification models such as$\boldsymbol{svm_classifier}()$,$\boldsymbol{naive_classifier}()$ and $\boldsymbol{nearest\_neighbor\_classifier}()$.

According to the nature of the datasets and the number of the data samples in the dataset a particular classifier can be chosen. After training the classifier the new upcoming samples can be classified.

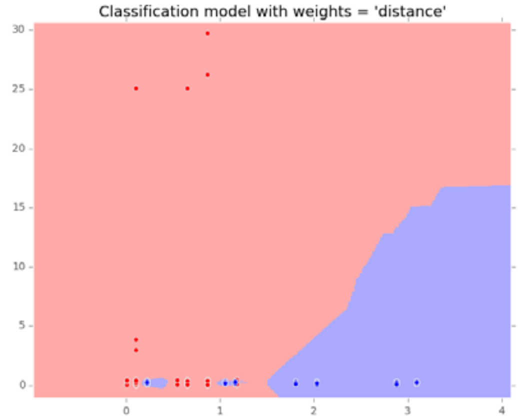


Figure 4.2(c) $INTEGRATED\_CLASSIFICATION()$ output2

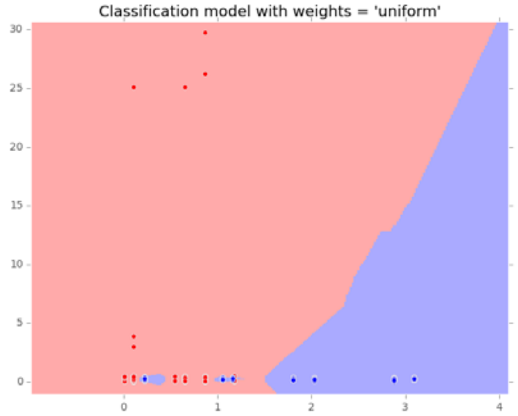


Figure 4.2(d) $INTEGRATED\_CLASSIFICATION()$

output3

V. RESULT AND ANALYSIS

In our experiments we have estimated the results by taking three factors into consideration that are-Accuracy, Precision, Recall and Computational Time. To do the comparison we have selected two well-known strategies and the results have shown that our strategy outperforms the other. Without any integration of metaheuristic with the integrated classifier module of our proposed framework, the data samples are classified based on the internally selected features by the algorithm. It does not guarantee an optimal subset of features. So, as the data samples increase the misclassification rate also increases. Our proposed framework resolves the classification problem by selecting a subset of top k features of the original dataset. In this, ACC-PSO first search for an optimal feature subset and based on that feature subset the classification model classifies the data samples. This Method reduces the computational time for the classification by keeping the Accuracy, Precision and Recall rate to a measurable level. Instead of standard PSO we have used a variant of PSO i.e. Rapid PSO for the purpose of feature selection and the results have shown that it finds an optimal feature subset within a less amount of time while maintain the measurable levels of other factors Accuracy, Precision and Recall.

The classification report of the classification model in the proposed work contains four major matrices- **precision, recall, f$_1$-score.**

- **Precision:** Precision of a classification model is the result relevancy. High precision rates indicates low false positive rate. So it is necessary to have a high precision rate for a good classification model. Precision (P) can be defined as the number of true positive (T_p) over the number of true positive plus the number of false positives (F_p)[20].

$$P = \frac{T_p}{T_p + F_p}$$

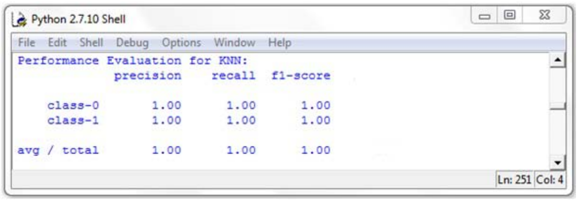
- **Recall:**Recall is a measure of how many truly relevant results are returned. High recall rate indicates low false negative rate. It should better to have high recall rate for good classification model.Recall(R) can be defined as the number of true positive (T_p) over the number of true positive plus the number of false negative(F_n)[20].

$$R = \frac{T_p}{T_p + F_n}$$

- **F1-score:** F1(score) can be defined as the harmonic mean of precision and recall [20].

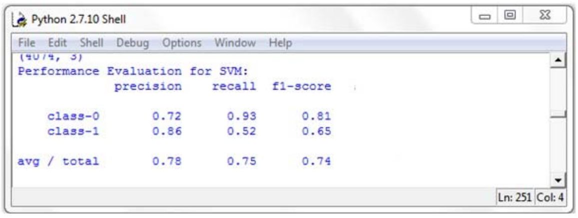
$$F1 = 2\frac{P \times R}{P + R}$$

The outputs of the classification is shown below-



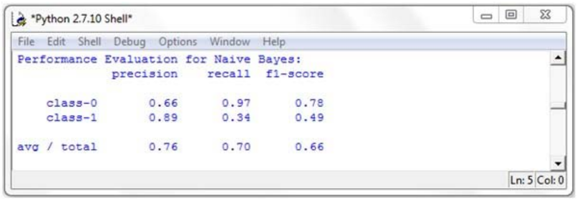
```
Performance Evaluation for KNN:
             precision    recall  f1-score

    class-0       1.00      1.00      1.00
    class-1       1.00      1.00      1.00

avg / total       1.00      1.00      1.00
```

Figure 5.1(a) Classification report for KNN



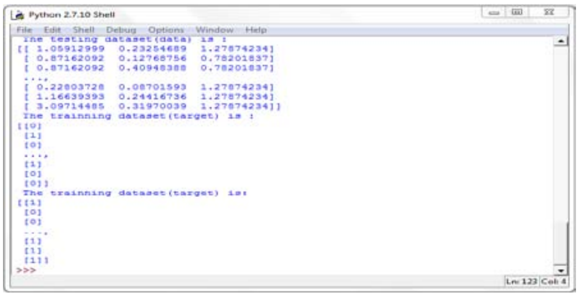
```
(4074, 3)
Performance Evaluation for SVM:
             precision    recall  f1-score

    class-0       0.72      0.93      0.81
    class-1       0.86      0.52      0.65

avg / total       0.78      0.75      0.74
```

Figure 5.1(b) Classification report for SVM



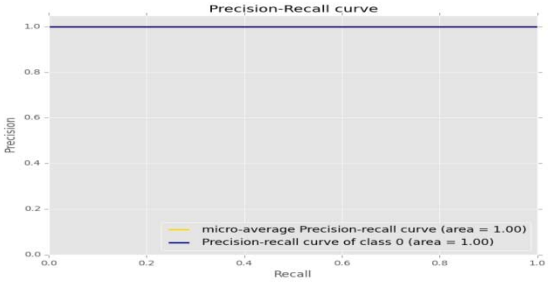
```
Performance Evaluation for Naive Bayes:
             precision    recall  f1-score

    class-0       0.66      0.97      0.78
    class-1       0.89      0.34      0.49

avg / total       0.76      0.70      0.66
```

Figure 5.1(c) Classification report for Naïve Bayes

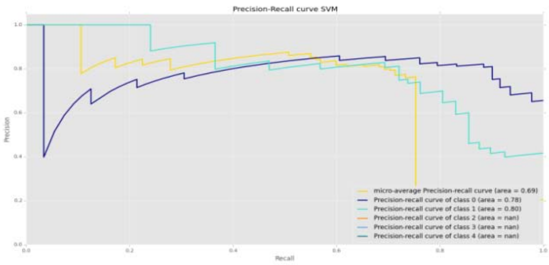


```
 The testing dataset(data) is :
[[ 1.05912999  0.23254689  1.27874234]
 [ 0.87162092  0.12768756  0.78201837]
 [ 0.87162092  0.40948388  0.78201837]
 ...,
 [ 0.22803728  0.08701593  1.27874234]
 [ 1.16639393  0.24416736  1.27874234]
 [ 3.09714485  0.31970039  1.27874234]]
 The trainning dataset(target) is :
[[0]
 [1]
 [0]
 ...,
 [1]
 [0]
 [0]]
 The trainning dataset(target) is:
[[1]
 [0]
 [0]
 ...,
 [1]
 [1]
 [1]]
>>>
```

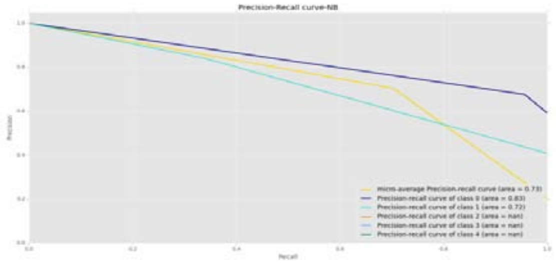
Figure 5.2 Split of the dataset after feature selection



Graph 5.3(a) Precision-Recall Curve for each of the class for KNN

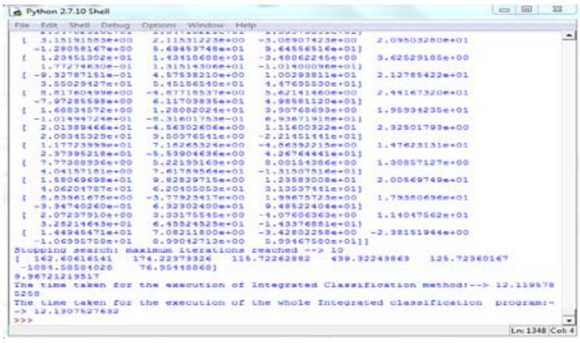


Graph 5.3(b) Precision-Recall Curve for each of the class for SVM



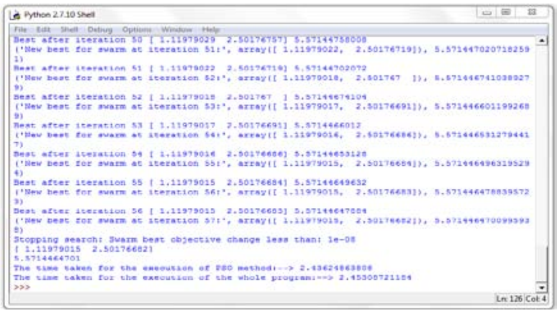
Graph 5.3(c) Precision-Recall Curve for each of the class for Naïve Bayes

In this section the execution time of the Integrated Classification, Classical-PSO-IC, IPSO-IC and Rapid PSO-IC has been compared and it is found that Rapid PSO is better than other PSO variants. All the codes had been executed stand alone for the reason. The following are the results generated from them which are showing the time taken.



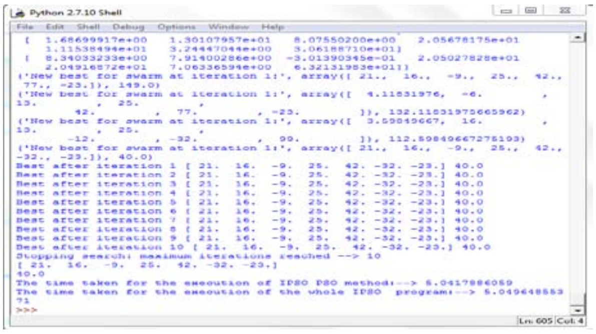
```
 [  3.15191583e+00   2.11531223e+00  -3.08907423e+00   2.09503280e+01
   -1.28058167e+00   5.69453748e+01   9.64556516e+01]
 [  1.23451302e+01   1.43418608e+01  -3.48062245e+00   3.62529185e+00
    1.77274630e-01   1.31514306e+01  -1.01400096e+01]
 [ -9.32787161e-01   4.57538210e+00   1.00293811e+01   2.12785422e+01
    3.55029427e+01   5.45106540e+01   4.47695530e+01]
 [  8.81760499e+00  -4.87718537e+00   5.62141460e+00   2.44167320e+01
   -7.97285598e+00   6.11703835e+01   4.98581120e+01]
 [  1.60834572e+00   1.28082024e+01   3.90768693e+00   1.95934235e+01
   -1.01494724e+01  -8.31601753e-01   6.93671918e+01]
 [  2.01389466e+01  -4.56302606e+00   1.11600322e+01   2.32501793e+00
    2.00345325e+01   9.50976541e+00  -2.21451441e+01]
 [  1.17723999e+01   7.18265324e+00  -4.86392215e+00   1.47623131e+01
    2.37395218e+01  -5.53904636e+00   4.26764441e+01]
 [  7.77308936e+00   5.22199169e+00   8.00154386e+00   1.30857127e+00
    4.04157181e+00   7.61789564e+01  -1.31507518e+01]
 [  1.58069698e+01   9.82829715e+00   1.23583008e+01   2.00569749e+01
    4.06204787e+01   6.20405053e+01   3.13537441e+01]
 [  8.83961678e+00  -3.77923417e+00   1.99675723e+00   1.79380696e+01
   -3.34740260e-01   6.92302400e+01   9.48522404e+01]
 [  2.07237910e+00   3.33175545e+00  -4.07606363e+00   1.14047562e+01
    3.28214643e+01   6.45524525e+01  -1.43376881e+01]
 [  1.44945471e+01   7.08211800e+00  -3.42802258e+00  -2.38151944e+00
   -1.06995750e+01   8.99042713e+00   5.99467500e+01]]
Stopping search: maximum iterations reached --> 10
[  162.60616141   174.22373326   115.72262882   439.32243863   125.72360167
 -1084.58584028    76.95448848]
9.96721219517
The time taken for the execution of Integrated Classification method:--> 12.119578
5258
The time taken for the execution of the whole Integrated classification  program:-
-> 12.1307527632
>>>
```

Figure 5.4(a) Execution time of the Integrated Classification



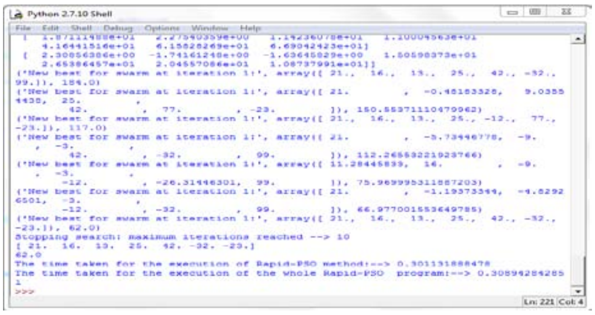
```
Best after iteration 50 [ 1.11979029  2.50176757] 5.57144758008
('New best for swarm at iteration 51:', array([ 1.11979022,  2.50176719]), 5.57144702071825
1)
Best after iteration 51 [ 1.11979022  2.50176719] 5.57144702072
('New best for swarm at iteration 52:', array([ 1.11979018,  2.501767  ]), 5.57144674103892
9)
Best after iteration 52 [ 1.11979018  2.501767  ] 5.57144674104
('New best for swarm at iteration 53:', array([ 1.11979017,  2.50176691]), 5.57144660119926
9)
Best after iteration 53 [ 1.11979017  2.50176691] 5.5714466012
('New best for swarm at iteration 54:', array([ 1.11979016,  2.50176686]), 5.57144653127944
7)
Best after iteration 54 [ 1.11979016  2.50176686] 5.57144653128
('New best for swarm at iteration 55:', array([ 1.11979015,  2.50176684]), 5.57144649631952
4)
Best after iteration 55 [ 1.11979015  2.50176684] 5.57144649632
('New best for swarm at iteration 56:', array([ 1.11979015,  2.50176683]), 5.57144647883957
3)
Best after iteration 56 [ 1.11979015  2.50176683] 5.57144647884
('New best for swarm at iteration 57:', array([ 1.11979015,  2.50176682]), 5.57144647009939
5)
Stopping search: Swarm best objective change less than: 1e-08
[ 1.11979015  2.50176682]
5.5714464701
The time taken for the execution of PSO method:--> 2.43624863808
The time taken for the execution of the whole program:--> 2.45308721154
>>>
```

Figure 5.4(b) Execution time of the Classical PSO



```
 [   1.68699917e+00   1.30107957e+01   8.07550200e+00   2.05678175e+01
     1.11538494e+01   3.24447044e+00   3.06188710e+01]
 [   8.34033233e+00   7.91400286e+00  -3.01390345e-01   2.05027828e+01
     2.04916872e+01   7.06336594e+00   6.32131953e+01]]
('New best for swarm at iteration 1:', array([ 21.,  16.,  -9.,  25.,  42.,
  77.,  -23.]), 149.0)
('New best for swarm at iteration 1:', array([  4.11831976,  -6.          ,
 13.          ,  25.          ,
        42.          ,  77.          ,  -23.          ]), 132.11831975665962)
('New best for swarm at iteration 1:', array([  3.59849667,  16.          ,
 13.          ,  25.          ,
       -12.          ,  -32.          ,  99.          ]), 112.59849667275193)
('New best for swarm at iteration 1:', array([ 21.,  16.,  -9.,  25.,  42.,
-32.,  -23.]), 40.0)
Best after iteration 1 [ 21.  16.  -9.  25.  42. -32. -23.] 40.0
Best after iteration 2 [ 21.  16.  -9.  25.  42. -32. -23.] 40.0
Best after iteration 3 [ 21.  16.  -9.  25.  42. -32. -23.] 40.0
Best after iteration 4 [ 21.  16.  -9.  25.  42. -32. -23.] 40.0
Best after iteration 5 [ 21.  16.  -9.  25.  42. -32. -23.] 40.0
Best after iteration 6 [ 21.  16.  -9.  25.  42. -32. -23.] 40.0
Best after iteration 7 [ 21.  16.  -9.  25.  42. -32. -23.] 40.0
Best after iteration 8 [ 21.  16.  -9.  25.  42. -32. -23.] 40.0
Best after iteration 9 [ 21.  16.  -9.  25.  42. -32. -23.] 40.0
Best after iteration 10 [ 21.  16.  -9.  25.  42. -32. -23.] 40.0
Stopping search: maximum iterations reached --> 10
[ 21.  16.  -9.  25.  42. -32. -23.]
40.0
The time taken for the execution of IPSO PSO method:--> 5.0417886059
The time taken for the execution of the whole IPSO  program:--> 5.049648553
71
>>>
```

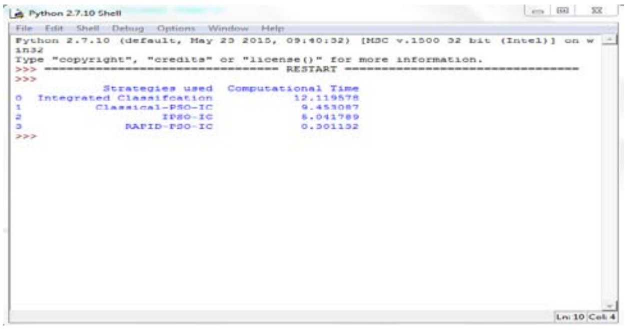
Figure 5.4(c) Execution time of the IPSO



```
 [  1.87111488e+01   2.27540359e+00   1.14236078e+01   1.10004563e+01
    4.16441516e+01   6.15828269e+01   6.69042423e+01]
 [  2.30856386e+00  -1.74161248e+00  -1.63645829e+00   1.50598373e+01
    2.65386457e+01   2.04557086e+01   1.08737991e+01]]
('New best for swarm at iteration 1:', array([ 21.,  16.,  13.,  25.,  42.,  -32.,
99.]), 184.0)
('New best for swarm at iteration 1:', array([ 21.          ,  -0.48183328,   9.0355
4438,  25.        ,
        42.        ,  77.        ,  -23.        ]), 150.55371110479962)
('New best for swarm at iteration 1:', array([ 21.,  16.,  13.,  25., -12.,  77.,
-23.]), 117.0)
('New best for swarm at iteration 1:', array([ 21.          ,  -5.73446778,  -9.
    ,  -3.        ,
        42.        ,  -32.        ,  99.        ]), 112.26558221923766)
('New best for swarm at iteration 1:', array([ 11.28445833,  16.        ,  -9.
    ,  -3.        ,
       -12.        ,  -26.31446301,  99.        ]), 75.969995311887203)
('New best for swarm at iteration 1:', array([ 21.          ,  -1.19373344,  -4.8292
6501,  -3.        ,
       -12.        ,  -32.        ,  99.        ]), 66.977001553649785)
('New best for swarm at iteration 1:', array([ 21.,  16.,  13.,  25.,  42.,  -32.,
-23.]), 62.0)
Stopping search: maximum iterations reached --> 10
[ 21.  16.  13.  25.  42. -32. -23.]
62.0
The time taken for the execution of Rapid-PSO method:--> 0.301131888478
The time taken for the execution of the whole Rapid-PSO  program:--> 0.30894284285
1
>>>
```

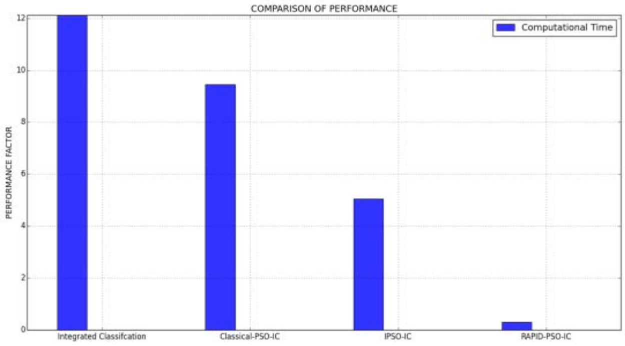
Figure 5.4(d) Execution time of the Rapid PSO

The table of comparison and graph are shown below



```
Python 2.7.10 (default, May 23 2015, 09:40:32) [MSC v.1500 32 bit (Intel)] on w
in32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
           Strategies used  Computational Time
0  Integrated Classifcation           12.119578
1          Classical-PSO-IC            9.453087
2                   IPSO-IC            5.041789
3              RAPID-PSO-IC            0.301132
>>>
```

Figure 5.3(c) Comparison table for the execution time



Graph 5.2 Comparison graph for execution time

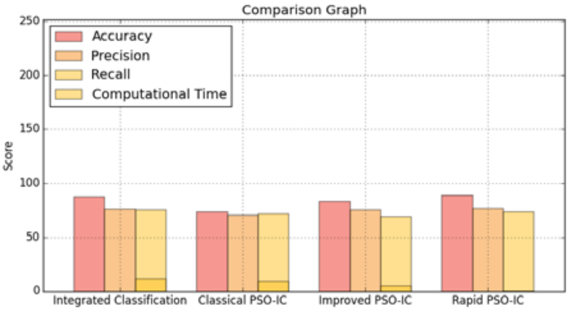
Table 5.1Comparison of various feature selection Techniques[7]

Technique	**Advantages**	**Disadvantages**	**Example**
Filter Approach	Scalability, Fast, Independent of the classifier	Not interaction with classifier, No Feature Dependencies	Euclidean Distance, i-test,χ^2 test etc.
Wrapper Approach	Interaction with Classifier, Easy to handle, Less Intensive	High Risk on over fitting, Computationally intensive, Classifier dependent on Feature selection	Genetic Algorithms,
Embedded Approach	Better computationally complexity, Interaction with Classifier,	Classifier dependent on Feature selection	Naïve Bayes, Decision Tree,

To compare the proposed classification model which is using accelerated PSO as feature selection method the python environment has been integrated with java environment. After this integration a real time comparison has been made and it proves the better performance of the proposed model. The following is the results generated from the integration

Table 5.1Result analysis in terms of accuracy, precision, recall and computational time

Strategies used	**Accuracy**	**Precision**	**Recall**	**Computational Time(Minutes)**
Integrated - Classification	88.0	76.512	75.6	12.1195
Classical PSO-IC	73.8	70.452	72.026	9.4530
IPSO-IC	83.6	75.567	69.780	5.0417
Rapid PSO-IC	89	76.6	73.87	0.3011



Graph 5.2 Comparisons of Classifiers

In Table 5.1 we can see that our proposed model outperforms the other existing method in computational time. It takes less amount of computational time than the other existing methods.

VI. CONCLUSION

Sentiment analysis is one of the most promising fields in many of the research communities in recent trends. All data that are being accumulated on the platforms like – Social Networking Site, Blogging and news websites are based on human emotions. So there should be a proper mechanism or model to justify the level of emotions of these data. Sentiment analysis comes under the category of classification problem which in turn is a machine learning problem. The proposed work deals with the above problem through a classification model which integrates a new metaheuristic optimization technique for feature selections. This technique is a variation of Particle Swarm Optimization called – Rapid-PSO. The results of the experiments show that the proposed model outperforms the other existing model which uses metaheuristics for the feature selection.

REFERENCES:

[1] S. Banker and R. Patel, “a Brief Review of Sentiment Analysis Methods,” *Int. J. Inf. Sci. Tech.*, vol. 62, no. 1, pp. 89–95, 2016.

[2] N. F. F. Da Silva, E. R. Hruschka, and E. R. Hruschka, “Tweet sentiment analysis with classifier ensembles,” *Decis. Support Syst.*, vol. 66, pp. 170–179, 2014.

[3] L. Marsh and C. Onof, “Stigmergic epistemology, stigmergic cognition,” *Cogn. Syst. Res.*, vol. 9, no. 1–2, pp. 136–149, 2008.

[4] W. Deng *et al.*, “An improved CACO algorithm based on adaptive method and multi-variant strategies,” *Soft Comput.*, vol. 19, no. 3, pp. 701–713, 2014.

[5] I. Guyon and A. Elisseeff, “An Introduction to Variable and Feature Selection,” *J. Mach. Learn. Res.*, vol. 3, no. 3, pp. 1157–1182, 2003.

[6] B. Xue, M. J. Zhang, and W. N. Browne, “Particle swarm optimization for feature selection in classification: a multi-objective approach,” *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1656–71, 2013.

[7] Y. Saeys, I. Inza, and P. Larra??aga, “A review of feature selection techniques in bioinformatics,” *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[8] “Sentiment Classifier | uClassify.” [Online]. Available: https://www.uclassify.com/browse/uclassify/sentiment?input=Text. [Accessed: 09-Dec-2017].

[9] “Text Classification and Sentiment Analysis – Ahmet Taspinar.” [Online]. Available: http://ataspinar.com/2015/11/16/text-classification-and-sentiment-analysis/. [Accessed: 11-Dec-2017].

[10] M. G. Smith and L. Bull, “Genetic programming with a genetic algorithm for feature construction and selection,” *Genet. Program. Evolvable Mach.*, vol. 6, no. 3, pp. 265–281, 2005.

[11] F. E. B. Otero, M. M. S. Silva, A. A. Freitas, and J. C. Nievola, “Genetic Programming for Attribute Construction in Data Mining,” *Genet. Program. Proc. EuroGP’2003*, vol. 2610, pp. 384–393, 2003.

[12] C. T. Tran, M. Zhang, P. Andreae, and B. Xue, “Improving performance for classification with incomplete data using wrapper-based feature selection,” *Evol. Intell.*, vol. 9, no. 3, pp. 81–94, 2016.

[13] G.-G. Wang, A. Hossein Gandomi, X.-S. Yang, and A. Hossein Alavi, “A novel improved accelerated particle swarm optimization algorithm for global numerical optimization,” *Eng. Comput.*, vol. 31, no. 7, pp. 1198–1220, 2014.

[14] W. Deng, R. Chen, B. He, Y. Liu, L. Yin, and J. Guo, “A novel two-stage hybrid swarm intelligence optimization algorithm and application,” *Soft Comput.*, vol. 16, no. 10, pp. 1707–1722, 2012.

[15] A. Mahanipour and H. Nezamabadi-pour, “Improved PSO-based feature construction algorithm using Feature Selection Methods,” *4. https://www.uclassify.com/browse/uclassify/sentiment?input=Text.*, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7940173/.

[16] D. p. Alonso M.G., “Particle Swarm Optimization (Pso): an Alternative Method for Composite Optimizati,” *10th World Congr. Struct. Multidiscip. Optim.*, pp. 1–10, 2013.

[17] J. Kennedy and R. Eberhart, “Particle swarm optimization,” *Neural Networks, 1995. Proceedings., IEEE Int. Conf.*, vol. 4, pp. 1942–1948 vol.4, 1995.

[18] B. F. De Souza, C. P. L. F. De Carvalho, R. Calvo, R. Porf, A. T. Sao-carlense, and S. Carlos, “Multiclass SVM Model Selection Using Particle Swarm Optimization,” *Proc. Sixth Int. Conf. Hybrid Intell. Syst.*, pp. 4–7, 2006.

[19] C. L. Chan and C. L. Chen, “A cautious PSO with conditional random,” *Expert Syst. Appl.*, vol. 42, no. 8, pp. 4120–4125, 2015.

[20] B. Liu, *Web Data Mining*. 2011.