# Analysis of Data using PSO model and Random forest

**[1]Mahesh Chintaman Ahire, [2]Pranali Tusharrao Sawant, [3]Ragini Pundalik Mahale, [4]Ms. T. S. Pawar**

[1]Student, [2]Student, [3]Student, [4]Guide
Department of Information Technology,
Nashik District Maratha Vidya Prasarak Samaj's
Karmaveer Adv. Baburao Ganpatrao Thakare College of Engineering, Nashik, India

***Abstract*: Big data is rising many technical challenges in today's world which affects both academic researches as well as IT sectors. The reason behind this is the streaming data and the data dimensionality. This was found that streaming data accumulates exponentially making traditional Methods to become infeasible during real time data mining. When it comes to mining high dimensional data, the search space from which an optimal feature subset is selected grows exponentially in size, leading to an large demand in computation. So, to solve this problem, a novel lightweight feature selection is proposed. The feature selection is designed particularly for mining streaming data on the fly, by using particle swarm optimization (PSO) type of swarm search that achieves enhanced analytical accuracy within reasonable processing time. In this project, a collection of Big Data with exceptionally large degree of dimensionality are put under test of our new feature selection algorithm for performance evaluation.**

***Index Terms*: PSO, Random forest, CCV, feature selection.**

---

## I. INTRODUCTION

Now a days there is tremendous increase in the data streams and it becomes difficult to handle and process that data using traditional techniques. The main challenges faced are the 3V challenges.They are Volume, Velocity and Variety.Volume for Storing, Processing and Analysis of Data,Variety for various types of data which comes in the form of structured or unstructured data from any source of incoming data and the last one is Velocity to handle the large amounts of data.As it is visible all of us that everything around is being digitised which made human life much easier. But due to this digitalisation of surrounding environment, lots of data get generated by these digital devices. This size of data collected is so much large that it could not be handled with traditional data mining and knowledge discovery algorithms. Along with size of data, the rate at which newly data generated is also increased tremendously. It was seen in research in 2016 that 90% digital data is generated in previous 2 years. Each time when fresh data arrives it makes big data in ate to a bigger data. The traditional algorithms needs to build the model and rerun it again and again. This led to the development of new breed algorithms. Hence, to overcome this we need to develop some efficient methods. By using PSO model and Random forest we can overcoming these limitation.Random Forest algorithm is most favoured and most strong supervised machine learning algorithm. It is efficient of performing Regression as well as Classification tasks.Some of the pros of random forest that it can handle the missing values and maintains accuracy for missing data.PSO simulates the behaviors of bird flocking. PSO does not have genetic operators like crossover and mutation. Particles update themselves with the internal velocity. They also have memory, which is important to the algorithm. The feature selection is designed particularly for mining streaming data on the fly, by using particle swarm optimization (PSO) type of swarm search that achieves enhanced analytical accuracy within reasonable processing time.

## II. MOTIVATION AND THE RELATED WORK

### *Motivation:*

The motive of the project is to upgrade the data mining technique with some recent technologies which enhances the efficiency of mining. Various studies and practical implementations are done on similar types of project. Various decision tree algorithm were used. But, decision tree is an traditional method and the result obtained from it is not so much precise. While the recent data mining techniques like random forest can be used and the output gained will be more precise while it will consume same computational power and time.

### *Related Work:*

Traditional data mining algorithm lack in precision, accuracy, requires large computational power and consumes lots of time for mining. We can mine the data using regular data mining algorithm, but due to the above problems, it becomes infeasible to the user to use them. Data mining algorithm such as Option Tree, Decision Tree, Hoeffding Tree, etc., successfully mines the data and the result are also obtained but the accuracy of the result is not precise as per the prediction of the user. This is so because of the uncertainty in the data sets. Normally, when we use a decision tree for data mining on large datasets, the result obtained is the combined result of the large set of data. But if the similar data sets are mined using the number of decision trees along with some different combination of feature subsets, then individual decision tree will have an output which might be different from outputs of other trees. The best result which has maximum number of occurrences is used as final result of data mining. The group of decision tree used can be collectively called as forest of trees. And this whole method is termed as Random Forest. To show that random forest is more efficient the decision three we used an open source data mining tool called as Weka (Waikato Environment for Knowledge Analysis). For our experimentation we used weather dataset which is readily available in Weka software. We uses two

classification algorithm i.e. decision tree and random forest. After the experiment we found that random forest performs much better than decision tree and thus accuracy of the mining gets increase.

## III. METHODOLOGY

***Data Set:***

This is a dataset that depicts sonar chirp returns bouncing off various surfaces. The 60 input variables are the strength of the returns at different angles. It is a binary classification problem that requires a model to differentiate rocks from metal cylinders.The dataset is first loaded, the string values converted to numeric and the output column is converted from strings to the integer values of 0 and 1. This is achieved with helper functions **load_csv()**, **str_column_to_float()** and **str_column_to_int()** to load and prepare the dataset.

***Random forest :***

In random forest we develop different trees rather than a solitary tree to classify a new object. Based on attributes each tree gives a classification and we save the tree votes for that class.The forest chooses the classification have maximum votes (overall trees in forest) and in case of regression it takes average of the outputs by different trees.When there are more trees in forest random classifier wont overfit the model. It also have power to handle large datasets with higher dimensionality.

Sonar data set is used in our model. In random forest algorithm we will use k-fold cross validation to estimate the performance of the learned model on unseen data. This means that we will construct and evaluate k models and estimate the performance as the mean model error. Classification accuracy will be utilized to assess each model. These behaviors are provided in the **cross_validation_split()**, **accuracy_metric()** and **valuate_algorithm()** helper functions.

```
# Split a dataset into k folds
def cross_validation_split(dataset, n_folds):
dataset_split = list()
dataset_copy = list(dataset)
fold_size = int(len(dataset) / n_folds)
for i in range(n_folds):
   fold = list()
   while len(fold) < fold_size:
      index =randrange(len(dataset_copy))
    fold.append(dataset_copy.pop(index))
   dataset_split.append(fold)
return dataset_split
```

We will also likewise utilize an execution of the Classification and Regression Trees (CART) algorithm adapted for bagging including the helper functions test_split() to split a dataset into groups, **gini_index()** to assess a split point, our altered **get_split()** function examined in the past step, **to_terminal()**, **split()** and **build_tree()** used to create a single decision tree, **predict()** to make a prediction with a decision tree, **subsample()** to make a subsample of the training dataset and **bagging_predict()** to make a prediction with a list of decision trees.

A new function name randome_forest is developed that first create a list of decision tree from subsamples of the training dataset and then uses them to make predictions.

```
# Random Forest Algorithm
def random_forest(train, test, max_depth, min_size, sample_size, n_trees, n_features):
      trees = list()
      for i in range(n_trees):
         sample = subsample(train, sample_size)
         tree = build_tree(sample, max_depth, min_size, n_features)
         trees.append(tree)
      predictions = [bagging_predict(trees, row) for row in test]
      return(predictions)
```

As we know, the key difference between Random Forest and bagged decision trees is the one small change to the way that trees are created, here in the get_split() function.

***PSO(Particle Swarm Optimisation)***

Particle swarm optimization is a population-based stochastic optimization technique inspired by the social behavior of bird flocking or fish schooling, particularly useful for optimization problems. A swarm comprises of several particles where every molecule monitors of its own attributes. The attributes of any particle in the swarm include its present position as given by the n-dimensional vector and the present velocity of the particle to keep track of the speed and direction in which the particle is currently

moving. Each particle also has a current fitness value, obtained by evaluating the fitness function at the particle's current position.PSO is initialized with a group of random particles(solution) and then searches for optima by updating generations. The proposed feature selection algorithm is given below :

1. *Initialize Population*
2. *WHILE (Stopping criterion is not met)*
   a. *FOR p =1 to number of particles*
      i. *Select attributes*
      ii. *Separate Training data and Test data using k-fold cross validation*
      iii. *Train on Training data*
      iv. *Classify using Test data*
      v. *Store Intrusion detection rate in an array*
   b. *NEXT p*
3. *Update particle's velocity and position*
4. *NEXT generation until stopping criterion*

## IV. RESULT

1. The result achieved by sonar data set was same as we expected. The result evaluated in two parts, traditional model and proposed model. In traditional model, we used standard decision tree and random forest. The performance achieved by them was similar and it was 69.8%.

   While in proposed model, optimisation operation was iterated for 100 times. Result obtained by using PSO with Decision tree was 66.7% and with Random Forest was 77.8%. The best cost achieved with Decision Tree was 0.12. Optimiser was able to select 39 best features among 60. Similarly , best cost achieved with Random forest was 0.144. Optimiser was able to select 31 best features.

2. The same optimisation was done with pseudo randomly generated dataset. It consisted 15 features. In traditional model, the performance achieved using standard decision tree and random forest were 60% and 56.7%.

   In proposed model, after optimisation using PSO, result obtained with Decision tree was 43.3% and with Random Forest, it was 63.33%. The best cost achieved with Decision Tree was 0.259. Optimiser was able to select 12 best features among 15. Similarly , best cost achieved with Random forest was 0.203. Optimiser was able to select 8 best features.

## V. CONCLUSION

The proposed system was to develop a classifier model using Random forest instead of decision tree along with PSO optimiser. This is so because it was seen that a single decision tree was not able to build a classier model which was more of the precise. But due to the usage of random forest it was possible to train the model with adequate subsets of feature which resulted in better accuracy, precision of prediction and this result was selected for next iteration. And finally ending tree was resulted best tree for the model building to train the model with number of decision trees. This trees were using different subsets of feature for classifier model building.

## REFERENCES

[1] Kennedy, J.; Eberhart, R. (1995). "Particle Swarm Optimization". Proceedings of IEEE International Conference on Neural Networks. IV. pp. 1942–1948.
[2] Shi, Y.; Eberhart, R.C. (1998). "A modified particle swarm optimizer". Proceedings of IEEE International Conference on Evolutionary Computation.
[3] Kennedy, J. (1997). "The particle swarm: social adaptation of knowledge". Proceedings of IEEE International Conference on Evolutionary Computation.
[4] Kennedy, J.; Eberhart, R.C. (2001). Swarm Intelligence. Morgan Kaufmann.
[5] Poli, R. (2007). "An analysis of publications on particle swarm optimisation applications" (PDF). Technical Report CSM-469.
[6] Poli, R. (2008). "Analysis of the publications on the applications of particle swarm optimisation" (PDF). Journal of Artificial Evolution and Applications.