**Feature Selection Optimization using Binary Jaya Algorithm**

Objective function: Area Under the Curve (AUC) Score

Table of Contents

[Abstract 2](#_Toc57469496)

[Introduction 2](#_Toc57469497)

[Dataset 2](#_Toc57469498)

[Dataset-1: Musk 2](#_Toc57469499)

[Dataset-2: Madelon 2](#_Toc57469500)

[List of Symbols 2](#_Toc57469501)

[Classifiers 3](#_Toc57469502)

[Algorithm 3](#_Toc57469503)

[Step-1: Initialize Jaya Algorithm Parameters 3](#_Toc57469504)

[Step-2: Define Content 4](#_Toc57469505)

[Datasets 4](#_Toc57469506)

[Objective Function 4](#_Toc57469507)

[S-Function (Sigmoid Function) 5](#_Toc57469508)

[Classifier 7](#_Toc57469509)

[Step-3: Execute Binary Jaya Algorithm 7](#_Toc57469510)

[Step-A: Generate Binary Population 7](#_Toc57469511)

[Step-B: Determine Best and Worse Solutions 7](#_Toc57469512)

[Step-C: Improvement Process 8](#_Toc57469513)

[Step-D: Selection Process 8](#_Toc57469514)

[Step-E: Stop Criteria 8](#_Toc57469515)

[Algorithm 9](#_Toc57469516)

[Flow Diagram 10](#_Toc57469517)

[Results and Conclusion 11](#_Toc57469518)

[References 13](#_Toc57469519)

Figures and Tables

[Figure 1: S-shaped function for converting objective function output as a binary number 5](#_Toc57469520)

[Table 1: Musk Dataset Statistics 2](#_Toc57469521)

[Table 2: Madelon Dataset Statistics 2](#_Toc57469522)

[Table 3: Abbreviations 2](#_Toc57469523)

[Table 4: Nomenclature 3](#_Toc57469524)

[Table 5: Classifiers 3](#_Toc57469525)

# Abstract

The purpose of this document is to prepare a step by step guide for Ajay. The algorithm includes binary Jaya with the Area Under the Curve (AUC) score as a feature function. These algorithms are used to optimized the feature selection process and improve the overall accuracy of classification results using Naive Bayes (NB), K-Nearest Neighbour (KNN), LDA, and Regression Tree (RT).

# Introduction

The document contains the following:

* Datasets Musk and Madelon.
* Classifiers: NB, KNN, LDS, RT.
* Our Approach: Binary Jaya algorithm with AUC score as a function.
* Results

# Dataset

Two datasets are used for classification purposes. Both datasets are available at <https://archive.ics.uci.edu/ml/datasets>

## Dataset-1: Musk

Musk dataset statistics are shown in the table: 1.

This dataset is available at <https://archive.ics.uci.edu/ml/datasets/Musk+(Version+2)>.

## Dataset-2: Madelon

Madelon dataset statistics are shown in the table: 2:

This dataset is available at <https://archive.ics.uci.edu/ml/datasets/Madelon>.

Table 1: Musk Dataset Statistics

|  |  |  |
| --- | --- | --- |
| Content | Training Dataset | Test Dataset |
| Number of Features [] | 168 | 168 |
| Number of Measurements [] | 6598 |  |
| Number of Measurements [] |  | 476 |

Table 2: Madelon Dataset Statistics

|  |  |  |
| --- | --- | --- |
| Content | Training Dataset | Test Dataset |
| Number of Features [] | 500 | 500 |
| Number of Measurements [] | 2000 |  |
| Number of Measurements [] |  | 1800 |

# List of Symbols

Table : Abbreviations

|  |  |
| --- | --- |
| AUC Score | Area Under the ROC Curve Score |
| KNN | K-Nearest Neighbour |
| RT | Regression Tree |
| LDA | Linear Discriminant Analysis |
| FS | Feature Selection |

Table : Nomenclature

|  |  |
| --- | --- |
| P | Population Size |
| D | Minimum Number of Features |
| Random D | Length of a feature set |
| Fx | Objective Function |
| MaxIter | Maximum number of Iterations |
| S | Number of iterations |
| P | One population |
| X | Binary Population Set |
|  | Features of Musk Dataset |
|  | Features of Madelon Dataset |
|  | Number of training measurements of Musk dataset |
|  | Number of training measurements of Madelon dataset |
|  | Number of test measurements of Musk dataset |
|  | Number of test measurements of Madelon dataset |

# Classifiers

Four classifiers are used in this project with their default parameters are shown in the table below:

Table : Classifiers

|  |  |
| --- | --- |
| **Classifiers** | **Parameters** |
| **KNN** | Algorithm = ‘auto’, leaf\_size = 30, metric = ‘minkowski’  N\_neighbors = 10 , p = 2, weights = ‘uniform’ |
| **RT** | Criterion = ‘mse’ , max\_depth = None, min\_samples\_leaf = 1 , random\_state = None,  splitter = ‘best’ |
| **NB** | Priors = None, var\_smoothing = , tol = 0.0001,  store\_covariance = False |
| **LDA** | n\_components = None, solver = svd, tol = 0.0001,  store\_covariance = False |

# Algorithm

This section will explain the proposed algorithm step by step, along with the code.

## Step-1: Initialize Jaya Algorithm Parameters

Four parameters need to be initialized.

1. Population Size **(P)**: For Jaya algorithm execution, a specific number of population is required. The population size was *10*.
2. Number of Executions **(E)**: For result-averaging, . (the number of times you want to run the complete process).
3. Maximum Number of Iteration **(MaxIter)**: For the Jaya Algorithm number of iterations are required.
4. The minimum number of features **(D)**: Features are dimensions. In my opinion, it is vital to consider a certain number of dimensions that accommodate 80-90 % of the variance.

## Step-2: Define Content

Once parameters are initialized, it is essential to define the following:

### Datasets

For training purposes, two sample datasets are used. Musk training dataset below:

And Musk test dataset below:

2

Where and as shown in Table 1.

Similarly, Madelon training and test datasets are shown below:

And Madelon test dataset above:

4

Where and as shown in Table 2

### Objective Function

For analysis purposes, two objective functions are in place.

#### Error Function

This objective function aims to select the optimal subset of the features that minimize the fitness function of the selected classification models. The error is computed as the difference between the actual output, and the selected model estimation is presented by eq.5, where and is the number of testing observations for Madelon dataset and for Musk dataset. The fitness function values of the models are computed by dividing the summation of errors by the number of observations is presented by the eq.6

5

For each population where, , the objective function (eq.6) is calculated using eq.5.

6

#### Area Under the ROC Curve (AUC) Score

It is essential to understand the ROC curve before applying the AUC score. A ROC curve (receiver operating characteristic curve) shows a classification model's performance at all classification thresholds. This curve plots two parameters:

* True Positive Rate (TPR)

7

* False Positive Rate (FPR)

8

AUC measures the entire two-dimensional area underneath the entire ROC curve. AUC is desirable for the following two reasons:

* AUC is scale-invariant. It measures how well predictions are ranked, rather than their absolute values as done in eq.5.
* AUC is classification-threshold-invariant. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.

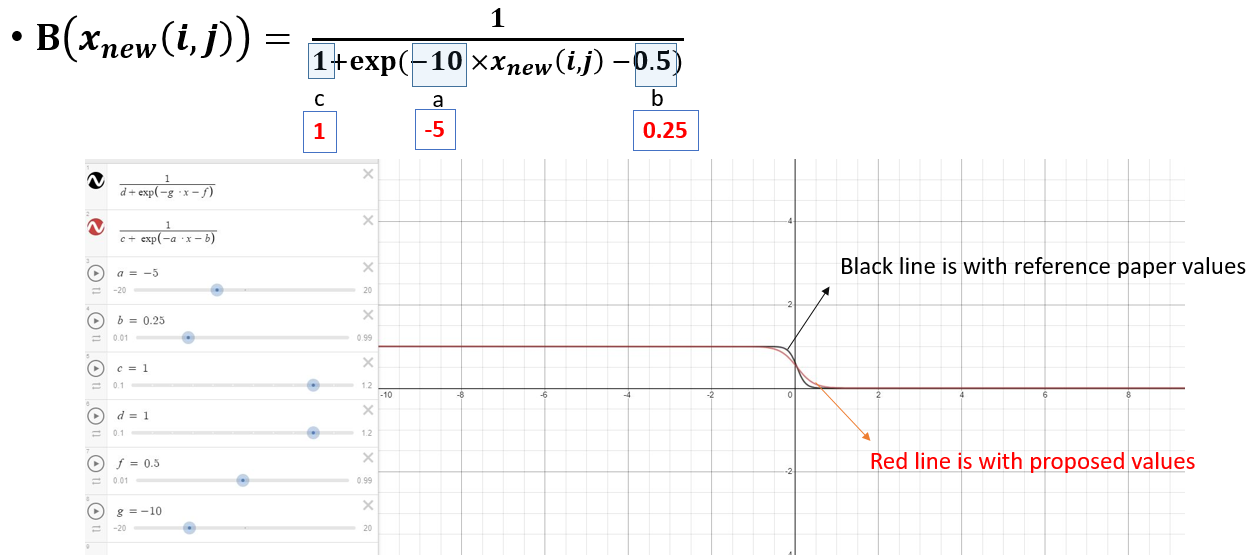
Moreover, the error function is based on the classifier's accuracy only, leading to an accuracy paradox. Secondly, when there is data imbalance, accuracy might increase due to only one class's influence. For comparison purposes, both objective functions are used.

### S-Function (Sigmoid Function)

The binary JAYA population is reconstructed using the JAYA algorithm using an objective function and selected features. However, the resulting values of the solutions are continuous. The sinusoidal transfer function (or S-shape transfer function (labelled as ‘’default’’ function) as drawn in Figure 1 is used to convert them to binary values (J & Eberhart, 1997). In practice, the new solution vector will be entered into eq.9 to calculate its binary vector.

9

Figure : S-shaped function for converting objective function output as a binary number



The following additional sigmoid functions are added:

**S- Shaped Functions**

|  |  |  |
| --- | --- | --- |
|  | Function | Visual |
| 1 | **Logistic Function** |  |
| 2 | **Hyperbolic function** |  |
| 3 | **Arctangent** |  |
| 4 | **gen\_logistic\_function** |  |
| 5 | algebraic function |  |
| 6 | gudermannian\_function |  |

### Feature Labels

Binary value from equation 9 further needs to label  into either ‘1’ or ‘0’. Two methods are proposed.

#### Random r

10

Here, rand() is the random number distributed between the range of 0 and 1.

#### Default r

### 

### Classifier

For objective function, we need to define a classifier. Any classifier can be selected from **Table 5**.)

## Step-3: Execute Binary Jaya Algorithm

### Step-A: Generate Binary Population

Construct the initial binary populations. In this step, each binary population is generated using a random selection of features. Later on, the fitness function cost of each generated population is calculated.

### Step-B: Determine Best and Worse Solutions

In this step, the best and the worse solutions are selected based on the objective function's value.

* **Objective function = error function:** The minimum value of the objective function will lead towards the best solution. The maximum value of the objective function will determine the worse value.
* **Objective function = AUC score:** The maximum value of the objective function will lead towards the best solution. The minimum value of the objective function will determine the worse value.

### Step-C: Improvement Process

In this step, every decision variable of every solution at each iteration is stochastically modified using the JAYA operator formulated in eq.11 (Awadallah, Al-Betar, Hammouri, & Alomari, 2020).

11

### Step-D: Selection Process

When the new solution X i is completely generated, it compares to the current one Xp stored in the population's pth position. Suppose the fitness value of Xnew is better than the fitness value of Xp. In that case, the new solution replaces the current one in the population.

This new fitness is compared with the old fitness.

* **Objective function = error function:** If the new fitness value is less than the old fitness value, its corresponding population is selected for the next iteration and otherwise dropped.
* **Objective function = AUC score:** If the new fitness value is more than the old fitness value, its corresponding population is selected for the next iteration and otherwise dropped.

### Step-E: Stop Criteria

Stop criterion. Steps B to D are repeated until either one of the following conditions is satisfied:

* the maximum number of iterations (i.e., Max\_itr) are satisfied.
* Only one population is left.

# Algorithm

Algorithm: The pseudo-code of the Proposed Algorithm

1. --------------- Initialize parameters and content ---------------------------
2. Initialize Jaya algorithm parameters. Population Size (**P**), Number of Executions (**E**), Maximum Iterations (**MaxIter**), Minimum number of Features (**D**)**.**
3. Load Datasets Train (Tr), Test (Te)
4. Initialize content. Objective Function, S-Function, Classifier
5. --------------- Initialize the binary population ------------------------------
6. Generate binary population and features.
7. Calculate
8. ---------------- Execution of Jaya Algorithm --------------------------------
9. **For** **do**
10. **For** m **do**
11. **if**  **do**
12. Determine the best solution in the population ()
13. Determine the worst solution in the population ()
14. **For** **do**
15. **For**  **do**


19. ------ binary conversion using S-Function ------

22. **End for**
23. Calculate
24. **End for**
25. --------- population update -------------
26. **If**  **do**
27. // update objective function
28. **Else** remove // remove population
29. *// update* ***P***
30. **Else** STOP **for** // for m
31. **End for**
32. **End for**
33. -------------------------------------- Optimal Subset of Features ------------------------------
34. Re-arrange Train and Test datasets according to optimal features
35. Classify

# Flow Diagram

JAYA ALGORITHM

YES

NO

New Training and Test Dataset

Classification

Optimal Subset of features

# Results and Conclusion

**MUSK DATASET – RESULTS**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FS Algorithms** | **NB** | | | **KNN** | | | **LDA** | | | **Regression Tree** | | |
| PR | Re | F1 | PR | Re | F1 | PR | Re | F1 | PR | Re | F1 |
| Without FS | 0.704 | 0.660 | 0.657 | 0.909 | 0.902 | 0.905 | 0.784 | 0.783 | 0.784 | 0.683 | 0.686 | 0.678 |
| **Test\_case-1**  Population = 10  Objective Function = **AUC**  Binary Conversion Function = **random\_r**  Sigmoid Function = **default** | 0.707 | 0.686 | 0.705 | **0.9073** | **0.900** | **0.9055** | **0.7844** | **0.7872** | **0.7878** | **0.7654** | **0.7627** | **0.7689** |
| **Test\_case-2**  Population = 10  Objective Function = **Error**  Binary Conversion Function = **random\_r**  Sigmoid Function = **default** | **0.7193** | **0.6979** | **0.7164** | 0.8882 | 0.8634 | 0.8739 | 0.7140 | 0.6830 | 0.705 | **0.7681** | **0.7718** | **0.7710** |
| **Test\_case-3**  Population = 10  Objective Function = **AUC**  Binary Conversion Function = **random\_r**  Sigmoid Function = **arctangent\_function** | **0.720** | **0.704** | **0.7206** | 0.885 | 0.8753 | 0.8824 | 0.725 | 0.713 | 0.7269 | **0.7670** | **0.7679** | **0.771** |
| **Test\_case-4**  Population = 10  Objective Function = **AUC**  Binary Conversion Function = **default\_r**  Sigmoid Function = **hyperbolic\_function** | **0.7125** | **0.691** | **0.710** | 0.888 | 0.876 | 0.884 | 0.744 | 0.739 | 0.747 | **0.7603** | **0.764** | **0.7626** |

**MADELON DATASET**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FS Algorithms** | **NB** | | | **KNN** | | | **LDA** | | | **Regression Tree** | | |
| PR | Re | F1 | PR | Re | F1 | PR | Re | F1 | PR | Re | F1 |
| Without FS | 0.496 | 0.496 | 0.495 | 0.504 | 0.503 | 0.496 | 0.513 | 0.513 | 0.513 | 0.498 | 0.498 | 0.498 |
| **Test\_case-1**  Population = 10  Objective Function = **AUC**  Binary Conversion Function = **random\_r**  Sigmoid Function = **default** | 0.528 | 0.528 | 0.528 | 0.5067 | 0.5072 | 0.5072 | 0.50 | 0.50 | 0.50 | 0.5078 | 0.5078 | 0.5078 |
| **Test\_case-2**  Population = 10  Objective Function = **Error**  Binary Conversion Function = **random\_r**  Sigmoid Function = **default** | 0.50 | 0.50 | 0.50 | 0.502 | 0.502 | 0.502 | 0.51 | 0.51 | 0.51 | 0.515 | 0.515` | 0.515 |
| **Test\_case-3**  Population = 10  Objective Function = **AUC**  Binary Conversion Function = **random\_r**  Sigmoid Function = **arctangent\_function** | 0.497 | 0.497 | 0.497 | 0.499 | 0.499 | 0.499 | 0.50 | 0.50 | 0.50 | 0.5033 | 0.5033 | 0.5033 |
| **Test\_case-4**  Population = 10  Objective Function = **AUC**  Binary Conversion Function = **default\_r**  Sigmoid Function = **hyperbolic\_function** | 0.498 | 0.498 | 0.498 | 0.502 | 0.502 | 0.502 | 0.502 | 0.502 | 0.50 | 0.5078 | 0.507 | 0.507 |

# Algorithm Rationale

This algorithm's ultimate purpose is to select the best features without compromising the classification results in terms of overall accuracy and individual class detection accuracy. Jaya algorithm is an optimization algorithm.

Results show improvement with the proposed algorithm. As mentioned earlier, the AUC score as an objective function can add logical value to the results, especially if the classes' data division is not balanced. **FOUR** test case scenarios with various parameters are tested against classification without feature extraction. It has been observed that all four classification algorithms are sensitive to the ‘Objective Function’, ‘Binary Conversion’ and ‘Sigmoid Function’. Interestingly these parameters do make a lot of difference in all classification frameworks.

The selection of its parameters and objective function plays a vital role in a classification problem. In the proposed algorithm, parameters and objective functions are selected based on maintaining the classifier's performance for overall accuracy and individual classes.

* **The minimum number of features:** it is essential to select at least a specific feature set, which covers 80-90% of data variance in a hyperspace. This is based upon the dimensionality reduction algorithm.
* **Area Under the Curve:** This objective function focused on a balanced ratio between True Positive Class detection versus True Negative Class detection. This is important as this function makes the complete feature selection process independent of any one-class impact. The value of the AUC score is higher when both classes detection rate is comparable. AUC score is a better option when the datasets are not balanced.
* **S-Function (Sigmoid Function): S-**function values are finalized using exhaustive search.

The result table above shows a definite improvement in overall classifiers performance with the above parameters and AUC score as an objective function compared to the classifier's performance with Error rate.

# References

Awadallah, M. A., Al-Betar, M. A., Hammouri, A. I., & Alomari, O. A. (2020). Binary JAYA Algorithm with Adaptive Mutation for Feature Selection. *Arabian Journal for Science and Engineering*.

Das, H., Naik, B., & Behera, H. (2020). A Jaya algorithm based wrapper method for optimal feature selection in supervised classification. *Journal of King Saud University - Computer and Information Sciences*.

J, K., & Eberhart. (1997). A discrete binary version of the particle swarm algorithm. *IEEE International Conference on Systems, Man and Cybernetics. Computational and Cybernetics and Simulations, 5*, 4104-4108.