The BireyselValue, a Proposed Method for Solving a Classification Problem

Deniz Dahman[1]

[1]Affiliation not available

March 27, 2024

Abstract

This paper *presents a new method for solving a classification problem*; the **BireyselValue** method assumes that the individual traits of a class help to classify an observation based on similarity measures. The method involves *three stages* to solve the classification problem: *the building stage, the training stage, and the prediction stage*. The first two stages accomplish *two key steps*: **firstly**, five parameters are used to transform any observation of size n variables into **six variables**; **secondly**, subsets of the individual traits of each class are created. As a result, the parameters, the subsets of the individual traits, and a scaled version of the training dataset are saved *as a predictive model*. Ultimately, the prediction stage uses the elements in the predictive model to transform the observations that are to be classified and of size n into the size of **six variables and to perform similarity measures** between the observation and the individual traits of class to make *the final prediction*. The experimental results obtained on 6 multiclass datasets from different domains showed that the proposed method is *efficient at solving classification problems*. Moreover, the method can *potentially be used for purposes other than solving a classification problem*.

Keywords: BireyselValue Method, Classification, Prediction, Dimension Reduction

1. Introduction

1.1. Principles of Classification

Classification is something that exists in nature in a very mysterious way. A class implies the collection of similar observations; the classification problem is the problem when an **observation** is to be classified in one of the classes based on the similarity of its characteristics with that of each class. Based on this implication, the characteristics of one or more groups of observations coincide and contrast with other observations. This points to the individuality of that observation and reflects its system profile.

1.2. The BireyselValue Method

This paper presents **a new method** for solving a classification problem based on the ***similarity*** between **the individual traits** of a given **class and the observation to be classified**. In particular, the method process involves **three key stages**: building, training, and prediction. In the building stage, four steps are involved in creating five parameters. In the training stage, seven steps are involved; consequently, subsets are formed, where is equal to the number of classes. Moreover, each observation from the training dataset is transformed into a shape of and placed into one of the subsets based on its original class. Each subset is

considered to represent the individual traits of a given class. Next, the five parameters, the subsets, and a scaled version of the training dataset were saved as a predictive model.

Finally, eleven steps are involved in the prediction stage. The five parameters and the scaled version of the training dataset in the saved predictive model are used to scale and then transform a given observation for which a class is sought. Next, a similarity check between the subsets, which are stored in the saved predictive model as the class individual trait, and the scaled, transformed observation are applied to make the final prediction.

The primary scenario for using the proposed method is as follows: given a training dataset with $m$observations, n variables, and $k \geq 2$ classes, the first two stages involved creating a predictive model. Finally, any given observation with n variables can be classified using the predictive model after the third stage is applied. **Fig**. (1) illustrates the workflow of the proposed method.

This paper is organized as follows: section () points to the motivation behind the name of the method and the conditions required to implement it. In addition, a training dataset is used as a showcase example to illustrate the steps of implementation. In section (), the paper introduces *in detail* the three stages and their steps using a showcase example. Furthermore, the mathematical formulations are discussed. Section () outlines the design of the experimental study by describing the experimental methodology and setup, the hyperparameter definitions, the evaluation measures and the results. Section () discusses the obtained results from different viewpoints. Finally, Section () concludes and presents the main outcomes of the study and some directions for future exploration in the research field.

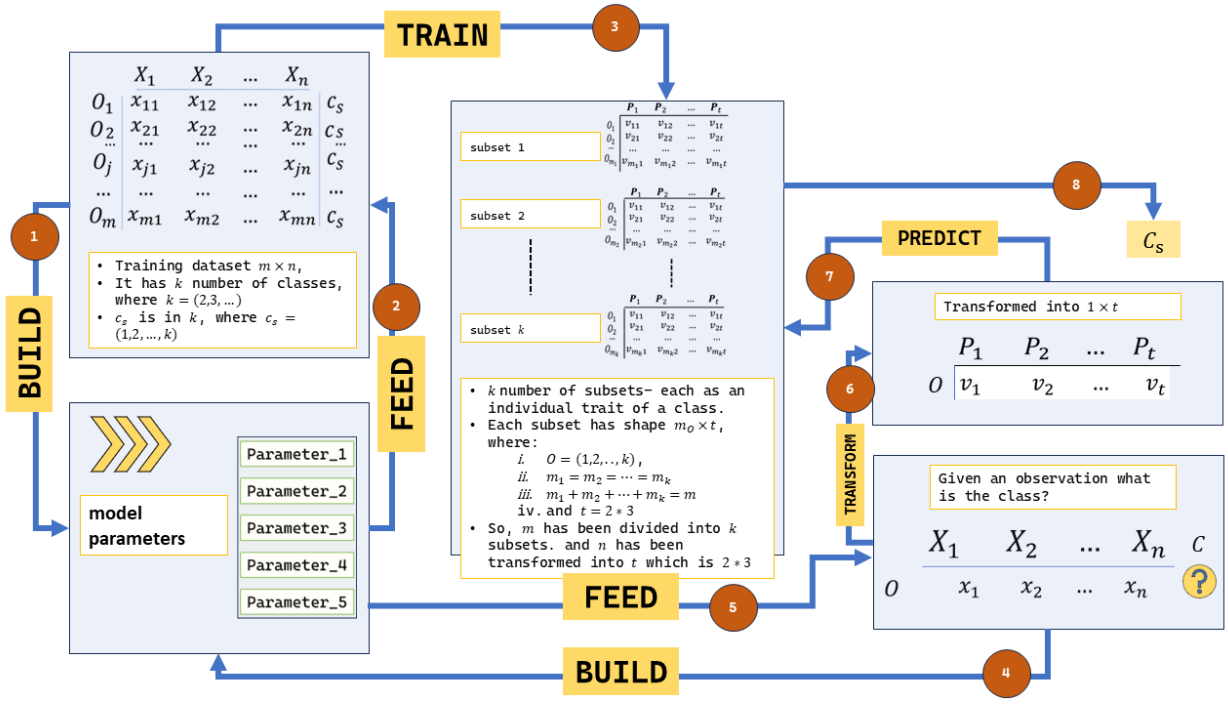


Figure 1: BireyselValue method workflow .

2. The BireyselValue name, conditions and showcase

2.1. Meaning of the name BireyselValue

The **similarity** check between *the individual traits* of a class and *the observation* to be classified, *outlined above in* , implies two things: the observations in the same class coincide with one another and have ***similar characteristics***; moreover, the individual traits of a class form **the backbone** of the BireyselValue method to predict the class of the given observation. To this end, those **individual traits** must be sought from observations that are in the same class. Since the ***personal*** characteristics of the observations *in the same class* are *the input* to form the ***individual*** traits of that class, the name **Bireysel** was used for this method. In the Turkish language, the word "Bireysel" has the same meaning as "personal" or "individual" as in the English language. The second part of the name "Value" is self-explanatory.

2.2. Conditions

Two conditions are required for the **BireyselValue** method to be employed:

2.2.1. The training dataset of size $m \times n$ must have $m \geq 80$, and $n \geq 2$.

2.2.2. The k classes must be ≥ 2. In addition, each class must have $m_{c_s} \geq 40$. m_{c_s} corresponds to the number of observations in a class.

2.3. Showcases

To demonstrate the use of the **BireyselValue** method, this paper will present an example of a *training dataset* with a size of $m = 120$ observations and $n = 5$ variables, which means that **600 entries** are measured. Furthermore, the training dataset has $k = 3$ classes as $C = (1, 2, 3)$.

3. Proposed Method

In this section, *the three stages* of the proposed method, named **BireyselValue**, are presented. The building stage is presented in , the training stage is presented in , and the prediction stage is presented in .

3.1 The Building Stage

In this stage, ***four*** steps are involved in creating five parameters; the sequence of the steps is as follows:

3.1.1. Step one

The norm of the training dataset is captured using **the Euclidean norm**, which is *the square root of the sum of every squared entry in the training dataset.* The result is a **scalar** value named **parameter_1**, which is referred to as v_{norm}. The calculation of the norm is formulated as follows:

$$\sqrt{\sum_{r=1}^{m}\sum_{a=1}^{n}\bar{x}_{ra}{}^{\circ 2}} \quad (1)$$

Next, **parameter_1** in Equation (1) is used to scale the training dataset; every entry x_{ra} in the training dataset is divided by **parameter_1**. As a result, the scaled version of the training dataset is referred to as **__ds**. **Fig**. (2) illustrates the outcome from this step using the showcase example in ().

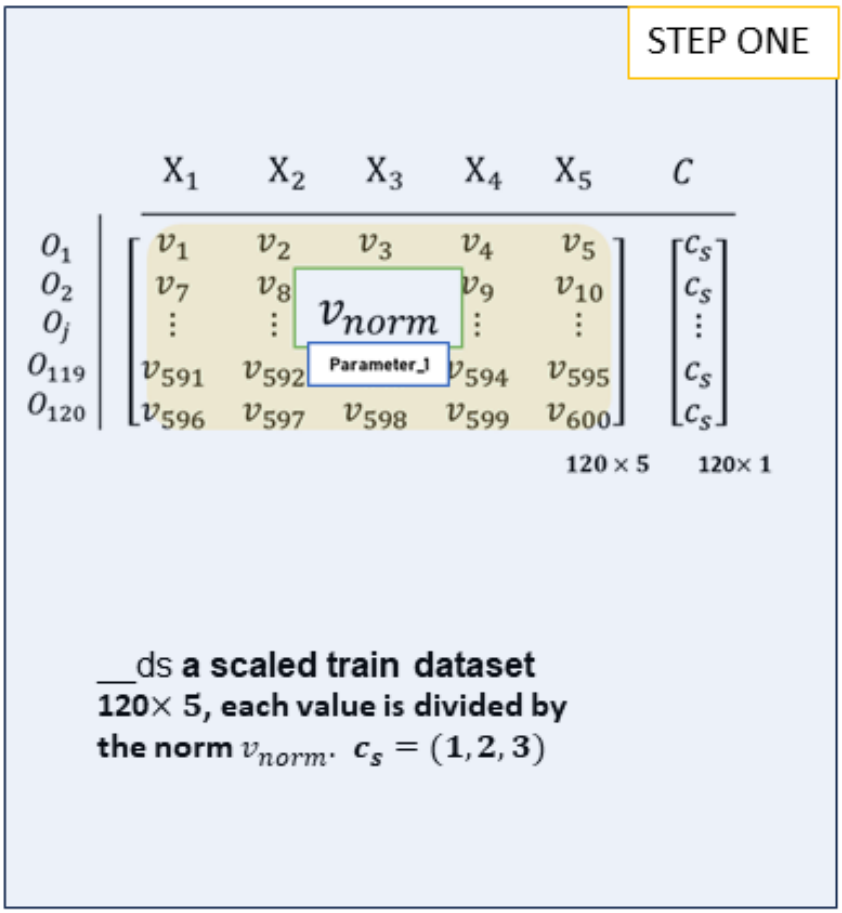


Figure 2: Building stage: step one outcomes: parameter_1 and _ds;

3.1.2. Step two

The main aim of this step is for every observation in **__ds** to have a vector representation of size k, where k is equal to the number of classes. Technically speaking, every observation in **__ds** is considered the **origin/center** of the **n-sphere** shape that has radius r; then, observations from the same or different classes, or perhaps none, will be inside the *n-sphere*. The number of observations m, the radius r, and the number of overlapping classes determine the number of observations inside the *n-sphere*. Fig. (3) illustrates this idea in 2D graphs using the showcase example in (), where two variables are chosen.

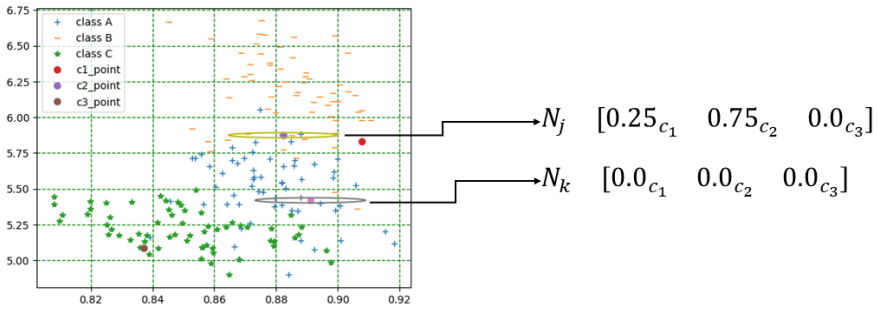


Figure 3: Observation from the _ds as the origin/center of 2-sphere and the percentage of observations from the same or different classes

The outcome of this step is as follows: a scalar value named **parameter_2**, referred to as r_{inner}; moreover, $m \times k$ matrix named **parameter_3**, referred to as **the neighbors' summary**; each row in the neighbors' summary corresponds to an observation in **__ds**; and the number of k columns is equal to the number of classes in **__ds**. **Parameters 2 and 3** are calculated in sequence as follows:

3.1.2.1. Fundamentally, a vector d of the **Euclidean distances** between each observation and other observations in **__ds** is calculated; the distance is *the square root of the squared difference between two observations* of size n, and it is formulated as follows:

$$\sqrt{^{-}(X_j - X_e)^{\circ 2}} (2)$$

Next, the value of **parameter_2** is captured; that is, the **midrange** of the vector d is tuned by the value $0.01 \leq l \leq 0.99$. The calculation of **parameter_2** is formulated as follows:

$$\frac{\min[?](d)+\max[?](d)}{2} \times l (3)$$

3.1.2.2. Finally, if *the Euclidean distance* in Equation (**??**) is less than or equal to **parameter_2** in Equation (**??**), the comparison is considered inside the **n-sphere**; the final result is a matrix, outlined above in (), named **parameter_3**. Fig. (4) illustrates the outcome from this step using the showcase example in ().

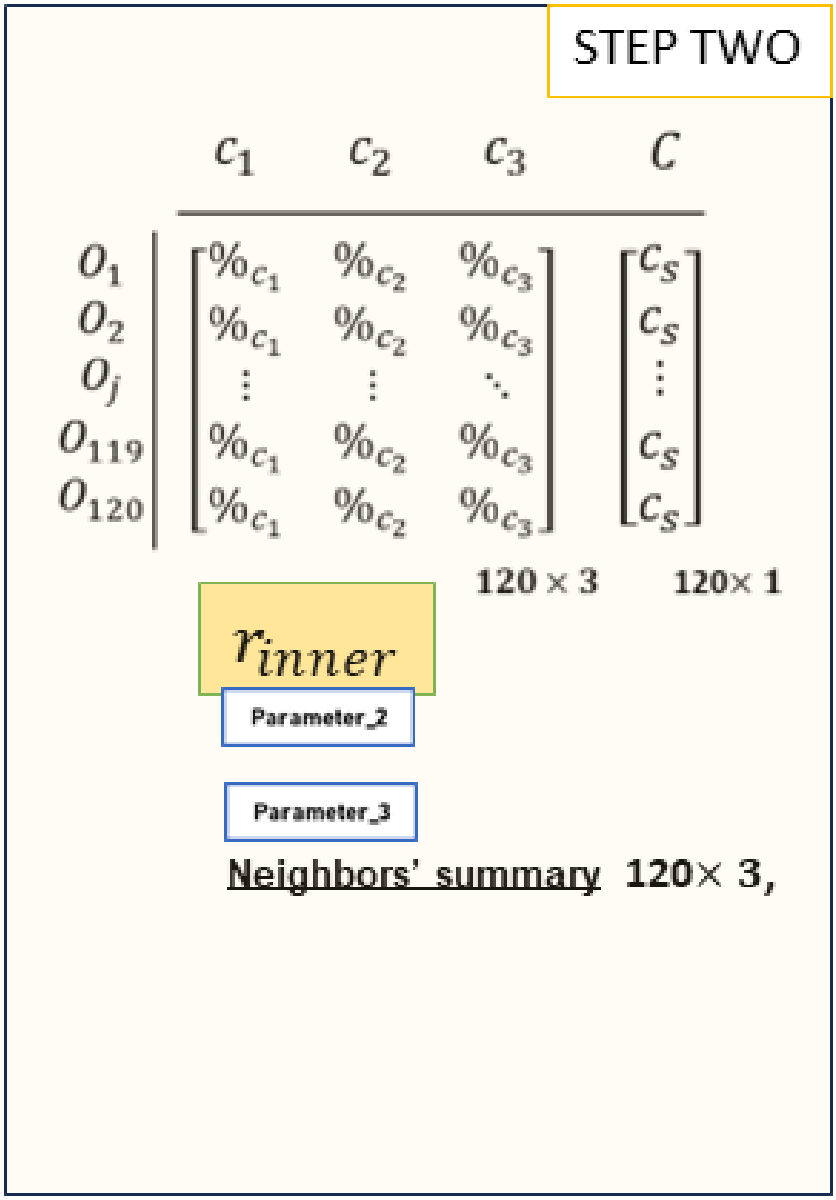


Figure 4: Building stage: step two outcomes: parameter_2 and parameter_3

3.1.3. Step three

The main aim of this step is for every class in __**ds** to have *an average value* representative. Technically speaking, every class has a number of observations, and each observation has n variables; on the basis of the assumption outlined in (), the sum of the n variables of each observation is sought. This is formulated as follows:

$$\sum_{i=1}^{n} x_i (4)$$

As a result, each class will have several sums from Equation (**??**); an average of the sums is calculated as

follows:

$$\frac{\min_{sum} + \max_{sum}}{2} (5)$$

Finally, the averages of each class collectively form a vector v of size k, where k is equal to the number of classes in **__ds**. Vector v is named **parameter_4** and is referred to as **avg vector**. Fig. (5) illustrates the outcome from this step using the showcase example in ().

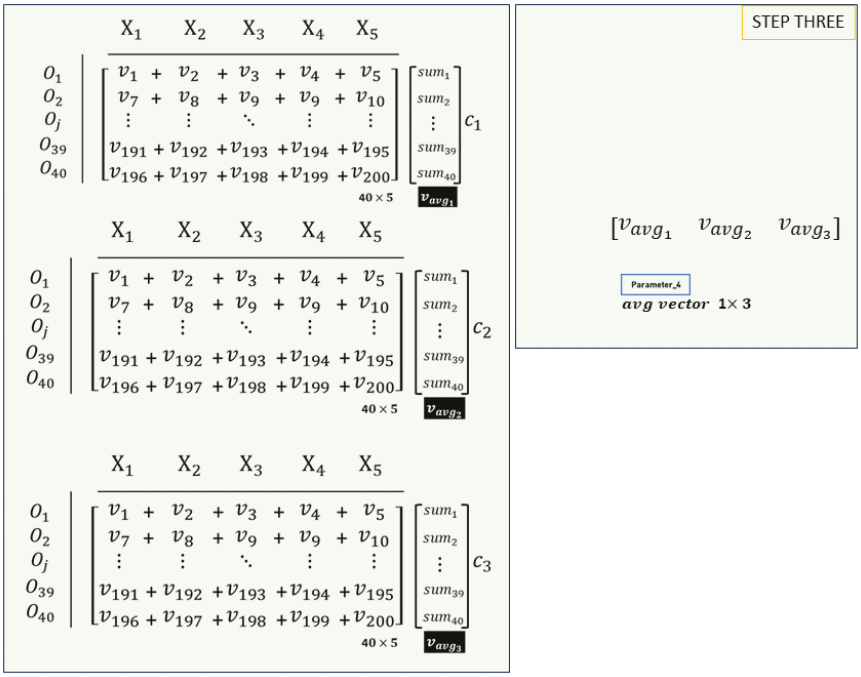


Figure 5: Budling stage: step three outcomes: parameter_4

3.1.4. Step four

The main aim of this step is for every class in **__ds** to have an **average error** value representative. Technically speaking, the **neighbors' summary** in () is split into k subsets, where k is equal to the number of classes in **__ds**; every subset contains observations belonging to the same class. *The maximum index value of each row is captured; if the maximum index value is not equal to the class index, the corresponding row observation in __ds is flagged.* As a result, each class contains the number of observations misclassified by the maximum number of neighbors in (). Next, the sum of the n variables of each observation is obtained. This is formulated as in Equation (**??**). In addition, an average of the sums is calculated as in Equation (**??**). Finally, **the average of the errors** of each class collectively forms a vector v of size k, where k is equal to the number of classes in **__ds**. Vector v is named **parameter_5** and is referred to as **err vector**. Fig. (6) illustrates the outcome from this step using the showcase example in ().

Fig. (7) illustrates the names and their references as the outcome from the building stage.

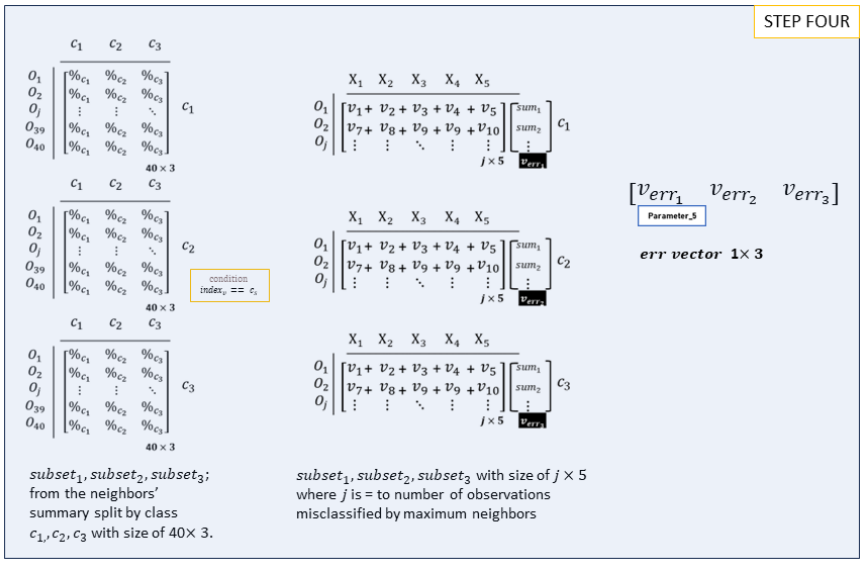


Figure 6: Budling stage: step three outcomes: parameter_4

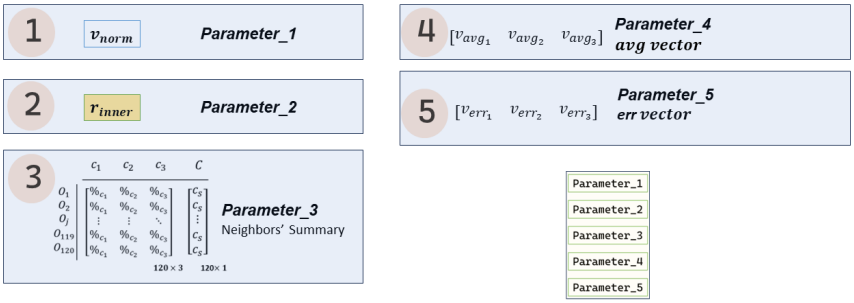


Figure 7: Building stage: step four outcomes: parameter_5

3.2 The Training Stage

In this stage, seven steps are include **transforming the shape** of each observation into the size (1×6) and *creating* **the individual traits** of each class, as outlined above in (); the sequence of the steps is as follows:

3.2.1. Step one:

The average sum of each observation in **__ds** is calculated as follows:

$$\sum_{i=1}^{n} x_i (6)$$

The result is a scalar value of each observation, named j_{sum}.

3.2.2. Step two:

The ***squared distance*** between j_{sum} in (**??**) and *the average vector* in () is captured; the result is a vector v of size k, named j_{avg}, where k is the number of classes in **__ds**.

3.2.3. Step three:

The ***squared distance*** betweenj_{sum} in (**??**) and *the error vector* in () is captured; the result is a vector v of size k, named j_{err}, where k is the number of classes in **__ds**.

3.2.4. Step four:

The corresponding **row observation** from the ***neighbors' summary*** in () is selected; the result is a vector v of size k, named j_{ns}, where k is the number of classes in **__ds**.

3.2.5. Step five:

Three vectors from (), (), and () are ***stacked and transposed***; the result is **a matrix** of size $(k \times 3)$ named j^T_{stack}. Fig. (8) illustrates the five steps in sequence using the showcase example in ().

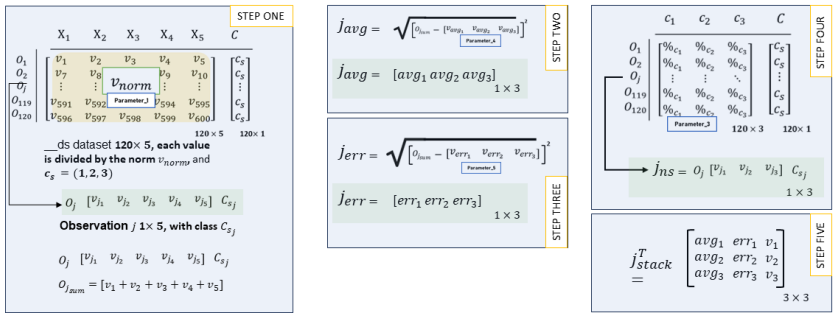


Figure 8: Training stage: step one, two, three, four, and five, in sequence

3.2.6. Step six:

The index of *the maximum* value of each column in () is captured. The result is a vector v of **size 3**, named j_{max}; **simultaneously**, the index of the *minimum* value of each column in () is captured. The result is a vector v of **size 3**, which is denoted as j_{min}.

3.2.7. Step seven:

The two vectors in () are **stacked**; the result is **a matrix** of **size** 2×3, named j_{img}, and **flattened** to a vector of **size** 1×6. As a result, each observation in **__ds** is ***transformed*** to a **size of** 1×6. Finally, **__ds** is *split into* k **subsets**; each subset represents **the individual traits of the class**, and k corresponds to the number of classes in **_ds**. Fig. (9) illustrates the previous two steps using the showcase example in ().

The __**ds** in (), the v_{norm} in (), the r_{inner} in (), the ***avg vector*** in (), the ***err vector*** in (), and ***the individual traits*** in () are **saved** as the ***predictive model***; Fig. (10) illustrates those elements.

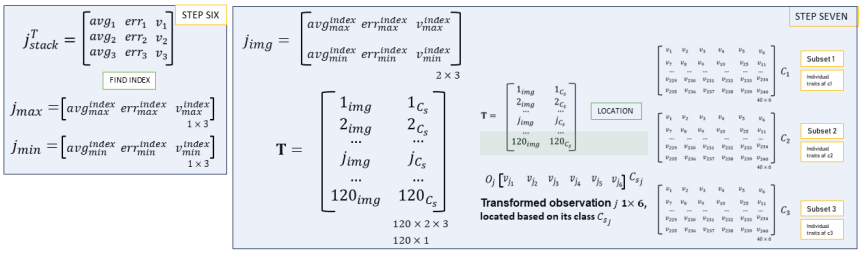


Figure 9: Training stage: step six, and seven, in sequence.

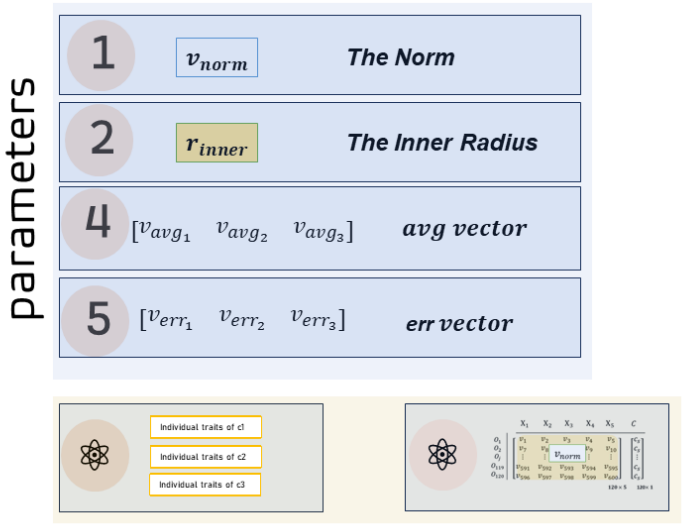


Figure 10: The predictive model from the building and the training stages, respectively; comprises, 4 parameters, the individual traits, and the __ds.

3.3. The Prediction Stage

In this stage, ***eleven steps*** are involved in **transforming** the **size** n of a given observation into the **size** 1×6 and ***predicting*** its class; the sequence of steps is as follows:

3.3.1. Step one:

The given observation is *scaled* by the v_{norm} in (). As a result, a scaled observation name _O of size n is used.

3.3.2. Step two:

Neighbors' summary created for the observation using the __**ds** in () and the r_{inner} in (). As a result, a vector v is named O_{ns} of size k, where k is equal to the number of classes in the __**ds**.

3.3.3. Step three:

The ***sum of*** n variables of the observation in () is captured. The result is a ***scalar*** named $O_{j_{sum}}$.

3.3.4. Step four:

The ***square root*** **difference** between $O_{j_{sum}}$ in () and the ***avg vector*** in () is calculated. The result is a vector v named O_{avg} of size k, where k is equal to the number of classes in the __**ds**.

3.3.5. Step five:

The ***square root difference*** between $O_{j_{sum}}$ in () and the ***err vector*** in () is calculated. The result is a vector v named O_{err} of size k, where k is equal to the number of classes in the __**ds**.

3.3.6. Step six:

Three vectors from (), (), and () are ***stacked and transposed***; the result is **a matrix** of **size** $k \times 3$ named O^T_{stack}.

3.3.7. Step seven:

The **index** of the *maximum* value of each column in () is captured. The result is a vector v of **size** 3, named $O_{\max}$; *simultaneously*, the **index** of the *minimum* value of each column in () is captured. The result is a vector v of **size** 3, which is denoted as $O_{\min}$. Fig. (11) illustrates ***the seven steps*** outlined above using the showcase example in ().

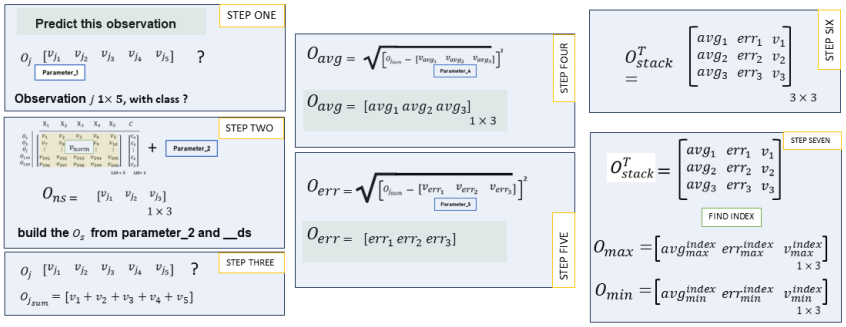


Figure 11: Prediction stage: step one, two, three, four, five, six, and seven in sequence

3.3.8. Step eight:

The **two vectors** in () are **stacked** and then **flattened** to a vector O_{img} of **size** 1×6.

3.3.9. Step nine:

A vector v of size P_{bp_a} is created with a size k, where k is equal to the number of classes in **__ds** in ().

3.3.10. Step ten:

A **similarity** check between O_{img} in () and *every observation* in the subset of **the individual traits** in () is performed, in every *match of similarity*, the corresponding **index** of b_{bp_a} in () **increases by 1**.

3.3.11. Step eleven:

The **maximum** value index in P_{bp_a} in () is **predicted** as the class.

Fig. (12) illustrates steps eight to eleven, outlined above, using the showcase example in ().

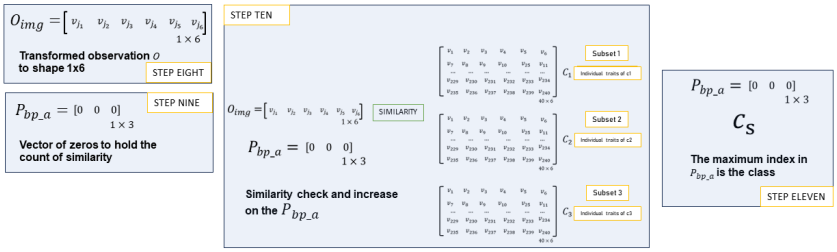


Figure 12: Prediction stage: step eight, nine, ten, and eleven in sequence

4. Experimental Evaluation

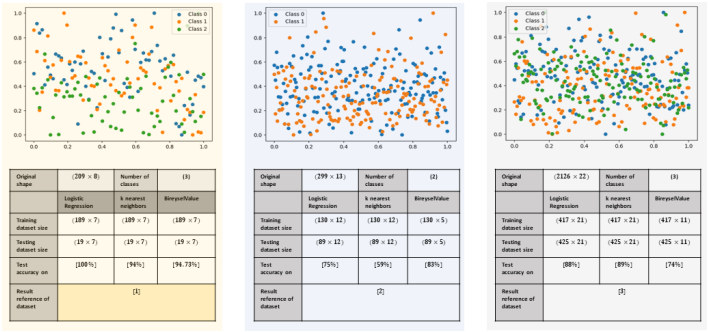
In this section, *the experimental evaluation* is presented, describing the data, the performance measures, the baseline algorithms, the results and other additional information.

4.1. Materials and Methods

4.1.1. Methods and Datasets:

Two classification methods, *logistic regression and the K-nearest neighbor* method, were chosen as the *baseline methods*; both were compared with the *BireyselValue* method in terms of *the performance metrics* mentioned below. Moreover, *six different multiclass datasets* were selected from several domains to evaluate the compression effect. The datasets were obtained from two repositories, as outlined in (). The datasets were randomly *split into training and testing sets*. Notably, none of the preprocessing, preparation, or cleaning steps were performed on the datasets. **However**, to satisfy the conditions for employing the BireyselValue

method outlined above in (2.2), a dedicated function from the BireyselValue package in Dahman (20241) was employed; as a result, a balanced training dataset was used for the three methods. The sizes of the original dataset, the numbers of classes, the sizes of the training and testing datasets, and the overall accuracy results are illustrated in Fig. (13) and Fig. (14). Notably, the scatter plots are created using two values: the first is the value from equation (**??**), and the second is the index of the observation. Overall, the scatter plot represents the overlapping classes of each dataset.



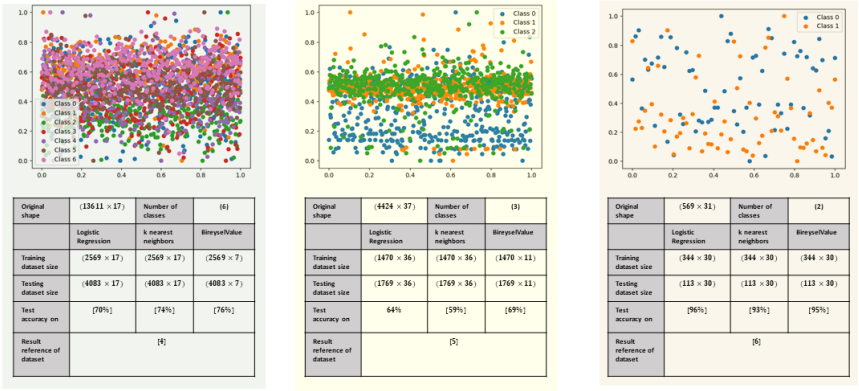
Original shape	(299 × 8)	Number of classes	(3)
	Logistic Regression	k nearest neighbors	BireyselValue
Training dataset size	(189 × 7)	(189 × 7)	(189 × 7)
Testing dataset size	(19 × 7)	(19 × 7)	(19 × 7)
Test accuracy on	[100%]	[94%]	[94.73%]
Result reference of dataset	[1]		

Original shape	(799 × 13)	Number of classes	(2)
	Logistic Regression	k nearest neighbors	BireyselValue
Training dataset size	(130 × 12)	(130 × 12)	(130 × 5)
Testing dataset size	(89 × 12)	(89 × 12)	(89 × 5)
Test accuracy on	[75%]	[59%]	[83%]
Result reference of dataset	[2]		

Original shape	(2126 × 22)	Number of classes	(3)
	Logistic Regression	k nearest neighbors	BireyselValue
Training dataset size	(417 × 21)	(417 × 21)	(417 × 11)
Testing dataset size	(425 × 21)	(425 × 21)	(425 × 11)
Test accuracy on	[88%]	[89%]	[74%]
Result reference of dataset	[3]		

1. (Charytanowicz, et al., 2010)
2. (Chicco & Jurman, 2020)
3. (Diogo Ayres, Bernardes, Garrido, Marques-de-Sá, & Pereira-Leite, 2000)

Figure 13: Result details on employing the BireyselValue method, Logistic Regression, and K nearest neighbors on 3 datasets



Original shape	(1361 1 × 17)	Number of classes	(6)
	Logistic Regression	k nearest neighbors	BireyselValue
Training dataset size	(2569 × 17)	(2569 × 17)	(2569 × 7)
Testing dataset size	(4083 × 17)	(4083 × 17)	(4083 × 7)
Test accuracy on	[70%]	[74%]	[76%]
Result reference of dataset	[4]		

Original shape	(4424 × 37)	Number of classes	(3)
	Logistic Regression	k nearest neighbors	BireyselValue
Training dataset size	(1470 × 36)	(1470 × 36)	(1470 × 11)
Testing dataset size	(1769 × 36)	(1769 × 36)	(1769 × 11)
Test accuracy on	64%	[59%]	[69%]
Result reference of dataset	[5]		

Original shape	(569 × 31)	Number of classes	(2)
	Logistic Regression	k nearest neighbors	BireyselValue
Training dataset size	(344 × 30)	(344 × 30)	(344 × 30)
Testing dataset size	(113 × 30)	(113 × 30)	(113 × 30)
Test accuracy on	[96%]	[93%]	[95%]
Result reference of dataset	[6]		

4. (Koklu & Ilker , 2020)
5. (Realinho, et al., 2021)
6. (Wolberg, et al., 1993)

Figure 14: Result details on employing the BireyselValue method, Logistic Regression, and K nearest neighbors on 3 datasets

4.1.2. Performance Measures:

The testing-based evaluation metrics (***precision, recall, and F1 score***) and accuracy (***overall, macro, and weighted average***) were used to evaluate the classification accuracy of the three methods applied to

the testing datasets.

4.1.3. Methods setup and execution environment:

The BireyselValue method has **one hyperparameter**, the *scalar of the radius*, which is used in the *building stage*, outlined above in (). For that purpose, values ranging from **0.1 to 0.3** are recommended. The range was chosen through experimentation, but in some datasets, such as Chicco Jurman (2020) and Koklu Özkan (2020), when using this range, *the accuracy measures after applying the prediction stage were very low*. For those datasets, the range was adjusted to values ranging from **0.05 to 0.09** to reach an acceptable accuracy. **In addition**, ***not all the variables*** n for the datasets Chicco Jurman (2020), Diogo Ayres (2000), Koklu Özkan (2020), and Martins . (2021) were selected; however, only **30-50%** of the variables were selected because, after some experimentation, the **BireyselValue** method performed much better with fewer variables. Notably, the selected variables are randomly chosen. A Python package from Dahman (20241) was used to construct the building, training and prediction stages, as outlined in (), (), and (), respectively. The documentation about the usage and the implementation steps are available Dahman (20242).

The k hyperparameter for *the K-nearest neighbor* method was fixed $k = 5$. The value was chosen through experimentation, and the best performance was observed at this value. **Similarly**, the *random state value* for the *logistic regression* method was fixed with a range from **5-16**, which varied depending on the dataset. For both methods, the Python package from **Scikit-learn** Pedregosa (2011) was used to perform the training and prediction steps.

Finally, the running environment was implemented on *a basic machine* containing a **5-core Intel(R) CPU (3.4 GHz-3.6 GHz) with 8 GB of RAM**.

4.2. Results

The performance metric results in () obtained using the three methods applied to the testing datasets are presented in this section. Each table corresponds to one of the datasets in (). **Notably**, **LR, KN, and BV** refer to *logistic regression, the K-nearest neighbor, and the BireyselValue*, respectively.

	precision			recall			F1-score			
	LR	KN	BV	LR	KN	BV	LR	KN	BV	Support
0	1	0.86	1	1	1	0.86	1	0.92	0.92	6
1	1	1	1	1	1	1	1	1	1	7
2	1	1	0.83	1	0.83	1	1	0.91	0.91	6
Overall accuracy							1	0.94	0.95	19
macro avg	1	0.95	0.94	1	0.94	0.95	1	0.94	0.94	19
weighted avg	1	0.95	0.96	1	0.95	0.95	1	0.95	0.95	19

Table 1: Results from the Charytanowicz . (2010) testing dataset

	precision			recall			F1-score			
	LR	KN	BV	LR	KN	BV	LR	KN	BV	Support
0	0.83	0.61	0.93	0.74	0.57	0.83	0.78	0.59	0.83	58
1	0.59	0.29	0.65	0.71	0.32	0.83	0.65	0.3	0.73	31
Overall accuracy							0.73	0.48	0.83	89
macro avg	0.71	0.45	0.79	0.73	0.45	0.83	0.71	0.45	0.8	89
weighted avg	0.75	0.5	0.85	0.73	0.48	0.83	0.73	0.49	0.84	89

Table 2: Results from the Chicco Jurman (2020) testing dataset

	precision			recall			F1-score			
	LR	KN	BV	LR	KN	BV	LR	KN	BV	Support
0	0.95	0.95	0.76	0.77	0.78	0.93	0.85	0.86	0.83	328
1	0.36	0.42	0.57	0.63	0.77	0.53	0.46	0.54	0.55	60
2	0.65	0.65	0.73	0.95	0.81	0.38	0.77	0.72	0.5	37
Overall accuracy							0.77	0.78	0.74	425
macro avg	0.65	0.95	0.69	0.78	0.94	0.61	0.69	0.71	0.63	425
weighted avg	0.84	0.95	0.73	0.77	0.95	0.74	0.79	0.8	0.72	425

Table 3: Results from Ayres-de Campos . (2000) testing dataset

	precision			recall			F1-score			
	LR	KN	BV	LR	KN	BV	LR	KN	BV	Support
0	0.85	0.81	0.54	0.82	0.74	0.49	0.83	0.78	0.51	1033
1	0.6	0.6	0.85	0.53	0.67	0.89	0.56	0.64	0.87	836
2	0.63	0.51	0.7	0.73	0.54	0.57	0.68	0.52	0.62	621
3	0.51	0.59	0.76	0.55	0.58	0.85	0.53	0.58	0.8	549
4	0.72	0.62	0.28	0.73	0.56	0.18	0.73	0.59	0.22	493
5	0.59	0.45	0.9	0.56	0.45	0.93	0.57	0.45	0.91	396
6	1	1	0.56	0.99	0.99	0.68	1	1	0.61	155
Overall accuracy							0.68	0.64	0.67	4083
macro avg	0.7	0.65	0.65	0.7	0.65	0.66	0.7	0.65	0.65	4083
weighted avg	0.68	0.64	0.66	0.68	0.64	0.67	0.68	0.64	0.66	4083

Table 4: Results from Koklu Özkan (2020) testing dataset

	precision			recall			F1-score			
	LR	KN	BV	LR	KN	BV	LR	KN	BV	Support
0	0.77	0.86	0.56	0.62	0.51	0.79	0.69	0.54	0.65	582
1	0.31	1	0.86	0.42	0.43	0.77	0.36	0.29	0.81	304
2	0.76	1	0.42	0.77	0.51	0.35	0.77	0.59	0.38	883
Overall accuracy							0.66	0.5	0.68	1769
macro avg	0.61	0.5	0.61	0.6	0.48	0.63	0.6	0.47	0.61	1769
weighted avg	0.69	0.58	0.7	0.66	0.58	0.68	0.67	0.52	0.68	1769

Table 5: Results from the Martins . (2021) testing dataset

	precision			recall			F1-score			
	LR	KN	BV	LR	KN	BV	LR	KN	BV	Support
0	0.96	0.97	0.97	1	0.95	0.95	0.98	0.96	0.96	73
1	1	0.9	0.9	0.93	0.95	0.95	0.96	0.93	0.92	40
Overall accuracy							0.97	0.95	0.95	113
macro avg	0.94	0.95	0.94	0.96	0.95	0.95	0.97	0.94	0.94	113
weighted avg	0.95	0.95	0.95	0.97	0.95	0.95	0.97	0.95	0.95	113

Table 6: Results from the Street . (1993) testing dataset

5. Discussion

In the overall analysis, spotting which method was the best using *overall accuracy*, which is the proportion of the correctly classified observations out of all the observations, it is possible to verify that the **BireyselValue** method performed at almost the same level of overall accuracy as the other two methods. **Indeed**, in two of the testing datasets, in particular Chicco Jurman (2020) (Table 2) and Martins . (2021) (Table 5), the **BireyselValue** outperformed the two methods, whereas with the other datasets, the **BireyselValue** was close to the most accurate.

All the testing datasets except for the first dataset Charytanowicz . (2010) (Tabel 1) are **imbalanced**. The *support column*, which refers to the number of actual occurrences of the class in the testing dataset, illustrates that. Therefore, the interpretation of the results from the performance measures in () should be carefully suggested. The two measures of the precision value, which are *the ratio of the true positive and the sum of the true positive and the false positive*, and the recall (or the sensitivity) value, which is *the ratio of the true positive and the sum of the true positive and the false negative*, are **intuitive** for the case of **a binary classification problem**; **however**, combining both together, *which is the value of the F1-score*, can be intuitive for the case of *a multiclass classification problem*. As a result, the **F1-score** is used to calculate *two different averages*: **(a)** the macro average and **(b)** the weighted average, where the first is calculated by taking the unweighted mean of all the per-class F1-scores, i.e., *this metric treats all the classes equally regardless of their support values*; the latter is calculated by *taking the mean of all the per-class F1-scores while considering each class's support value*. To this end, *the macro average is intuitive for balanced testing datasets, whereas the weighted average is intuitive for imbalanced testing datasets*.

The overall accuracy, the macro, and the weighted averages in (Tabel 1) show that the three methods perform at almost the **same level of accuracy**, even though the logistic regression appears to report 100% accuracy. However, **there is a reason to suggest**: the dataset Charytanowicz . (2010) is a common dataset, and the logistic regression method used for this experimentation was implemented Pedregosa (2011), which is a well-established library with an extensive level of resources; therefore, such accuracy could have been the result of hidden parameters that are tuned for such a common dataset.

The macro- and weighted averages results from (Tabel 2) show that the **BireyselValue** method **outperforms** the other methods. The overall accuracies were **73%, 48%**, and **83%** for the *LR, KN, and BV, respectively*. Similarly, the results from (Tabel 6) show that the performances of the three methods are very similar. Since both datasets are binary classification problems, *the precision and recall values are intuitive*; however, due to *the imbalance of both*, then the **F1-score** is the appropriate choice. *The F1-scores were much better captured by the proposed method for each class* in (Tabel 2); similarly, in (Tabel 6), the performances of the three methods were very similar. Furthermore, the level of accuracy captured by the **BireyselValue** in (Tabel 2) was based on the use of only **5 out of 12** variables.

The weighted average from (Tabel 4) shows that the three methods are very similar; **nonetheless**, the results from the baseline methods in compression with that one from the proposed method are **based on**

the 17 variables of the dataset, **whereby** the **BireyselValue** method **used only 7 out of the 17 variables**. Moreover, the dataset has *7 classes*, which emphasize the technical aspect mentioned in the previous paragraph.

Finally, (Tabel 3) shows that **the proposed method has the least performance** among the other two methods. Although the results of applying the three methods were within the same range of all the performance metrics, *one interpretation of the lowest performance achieved by the proposed method could be* that *the number of observations of each class* in the training datasets, i.e., the 139 observations of each class in the training dataset, **was insufficient** to provide additional information about the class. This interpretation is based on the *imbalanced distributions* of the three classes, which can be seen in the support column in (Table 3).

6. Conclusions and Future Works

In this paper, the discussion centers on **proposing a new method for solving a classification problem**. The **BireyselValue** method is based on **two key assumptions**: *first*, to transform both the observations in the training dataset and the one to be classified into variable **sizes of six**; *second*, to create an individual trait subset of each class. On the basis of these assumptions, *a zero vector* v of size k, where k is equal to the number of classes, is created; then, **a similarity check** between the observation to be classified and the individual trait subsets is implemented; every match will **increase the corresponding index in vector** v; and finally, based on the index of the maximum number in v, the **observation is classified**. This workflow is implemented in *three stages*: the **building stage**, which constructs five parameters; the **training stage**, which transforms the observations in the training dataset into the size of six variables and creates the individual trait subsets; and as a result, a predictive model is saved to perform the prediction stage. In the **prediction stage**, essentially, the observation to be classified is transformed into a size of six; then, a similarity check occurs; and finally, the prediction is made.

The experiments using **6 multiclass classification datasets** and **5 performance metrics** showed, in general, that the **BireyselValue** method can produce **competitive results** when compared to related works using classification methods. This finding **suggests that the proposed method is efficient** at solving classification problems. *Additionally*, the results showed that most of the accuracy of the proposed method was based on **30-50%** of the variables, **unlike** the other two methods. **This implies** two steps: **(a)** the proposed method can capture and then build an intuitive profile using a small number of variables, and **(b)** from a technical perspective, the mathematical equations that are used by the **BireyselValue** to build and train predictive models can dismiss the redundant (or the dependent) variables that are of no benefit to give any useful information about the class's traits.

Overall, the **BireyselValue** method is **primarily** used for solving classification problems. **However**, **another usage is possible**. An essential step in the training stage, as outlined in (), is to transform the observations in the training dataset into a variable size of six, i.e., the dataset of n variables, where n is any size, is r**educed to six dimensions**; as a result, the dataset of **size** $(m \times n)$ becomes $(m \times 6)$. Since this transformation is based on the individual traits of each class, t**he constructed six dimensions are most likely to contain the hidden characteristics of the observation**; as a result, this transformation could **address** the issue of **the curse of dimensionality** in **machine learning**. In addition, the new dataset can be used as the training dataset for other classification methods.

In future research, it is important to apply this method, mainly to achieve primary usage, on various types of datasets of several observations, variables, and classes; then, based on the accuracy reports, improvements are made. In addition, as outlined above, it is possible to use this method to address the curse of dimensionality in machine learning; such a claim is worthy of further research or study. To this end, this paper aims to be the first of a series of publications on research and studies on the improvements and usage of this method.

7. Declarations

7.1. Acknowledgements

- To the one who believed and gave with unconditional love.
- One who did not believe and gave pain and misery. As turns out to be my best fertilizer, pushing me to the limit, to learn more, and to be a better person.

7.2. Funding

The author declare that no funds, grants, or other support were received during the preparation of this manuscript.

7.3. Competing Interests

The author has no relevant financial or non-financial interests to disclose

7.4. Ethical conduction

The author declares no conflict of interest with respect to the ethical conduct terms and conditions.

7.5. Data Availability

The following data support the findings of this study:

1. The Seeds Dataset available in [**UCI Machine Learning Repository**] at https://doi.org/10.24432/C5H30K , reference Charytanowicz . (2010)
2. The Heart Failure Clinical Records dataset available in [**UCI Machine Learning Repository**] at https://doi.org/10.24432/C5Z89R , reference Chicco Jurman (2020)
3. The Fetal health Classification Dataset available in [**Kaggle**] at https://www.kaggle.com/andrewmvd/fetal-health-classification , reference Ayres-de Campos . (2000)
4. The Dry Bean Dataset available in [**UCI Machine Learning Repository**] at https://doi.org/10.24432/C50S4B , reference Koklu Özkan (2020)
5. Predict Students' Dropout and Academic Success Dataset available in [**UCI Machine Learning Repository**] at https://archive.ics.uci.edu/dataset/697/predict+students+dropout+and+academic+success , reference Martins . (2021)
6. Breast Cancer Wisconsin (Diagnostic) Dataset available in [**UCI Machine Learning Repository**] at https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic , reference Street . (1993)

References

Ayres$_{d}e_{C}ampos_{2}$000$Ayres - deCampos, D., Bernardes, J., Garrido, A., Marques - deS, J. Pereira - Leite, L.$ 2000.$Sisporto$2.0 : $Aprogramforautomatedanalysisofcardiotocograms Sisporto$2.0 : $Aprogramforautomatedanal$ 10.1002/1520-6661(200009/10)9:5¡311::aid-mfm12¿3.0.co;2-9

Charytanowicz2010CompleteGCCharytanowicz, M., Niewczas, J., Kulczycki, P., Kowalski, PA., Łukasik, S. Zak, S. 2010. Complete Gradient Clustering Algorithm for Features Analysis of X-Ray Images Complete Gradient Clustering Algorithm for Features Analysis of X-Ray Images.. https://api.semanticscholar.org/CorpusID:1570735

Chicco$_{2}$020$Chicco, D. Jurman, G.$ 2020$feb.Machinelearningcanpredictsurvivalofpatientswithheartfailurefromserumcre$ 10.1186/s12911-020-1023-5

dahman2024aDahman, D. 20241. The BireyselValue Algorithm Backage$_{t}estv$.1.0.0$TheBireyselValueAlgorithmBackage_{t}estv$.1.0.0.$PyPIThePythonPackageIndex$.

dahman2024Dahman, D. 20242. BireyselValue Project- Documentation BireyselValue Project- Documentation. GitHub. https://github.com/dahmansphi/bireyselvalue_v1

pubmedDiogo Ayres, JGAMdSJPLL., de-Campos; Bernardes. 2000. SisPorto 2.0: a program for automated analysis of cardiotocograms - PubMed SisPorto 2.0: a program for automated analysis of cardiotocograms - PubMed. Journal of maternal-fetal medicine(online). https://doi.org/10.1002/1520-6661(200009/10)9:5%3C311::AID-MFM12%3E3.0.CO;2-9 Accessed on Wed, March 20, 2024

Koklu2020MulticlassCOKoklu, M. Özkan, IA. 2020. Multiclass classification of dry beans using computer vision and machine learning techniques Multiclass classification of dry beans using computer vision and machine learning techniques. Comput. Electron. Agric.174105507. https://api.semanticscholar.org/CorpusID:219762890

Martins$_{2}$021$Martins, MV., Tolledo, D., Machado, J., Baptista, LMT. Realinho, V.$ 2021.$EarlyPredictionofstudent'sPerfo$ 10.1007/978-3-030-72657-7$_{1}$6

p2011Pedregosa, GGAMVTBGOBMPP., F; Varoquaux. 2011. Scikit-learn: Machine Learning in Python Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research122825–2830.

Street1993NuclearFEStreet, WN., Wolberg, WH. Mangasarian, OL. 1993. Nuclear feature extraction for breast tumor diagnosis Nuclear feature extraction for breast tumor diagnosis. Electronic imaging. Electronic imaging. https://api.semanticscholar.org/CorpusID:14922543