mathematics

MDPI

Article

Categorical Variable Mapping Considerations in Classification Problems: Protein Application

Gerardo Alfonso Perez * and Raquel Castillo

Biocomp Group, Institute of Advanced Materials (INAM), Universitat Jaume I, 12071 Castello, Spain
* Correspondence: ga284@cantab.net

Abstract: The mapping of categorical variables into numerical values is common in machine learning classification problems. This type of mapping is frequently performed in a relatively arbitrary manner. We present a series of four assumptions (tested numerically) regarding these mappings in the context of protein classification using amino acid information. This assumption involves the mapping of categorical variables into protein classification problems without the need to use approaches such as natural language process (NLP). The first three assumptions relate to equivalent mappings, and the fourth involves a comparable mapping using a proposed eigenvalue-based matrix representation of the amino acid chain. These assumptions were tested across a range of 23 different machine learning algorithms. It is shown that the numerical simulations are consistent with the presented assumptions, such as translation and permutations, and that the eigenvalue approach generates classifications that are statistically not different from the base case or that have higher mean values while at the same time providing some advantages such as having a fixed predetermined dimensions regardless of the size of the analyzed protein. This approach generated an accuracy of 83.25%. An optimization algorithm is also presented that selects an appropriate number of neurons in an artificial neural network applied to the above-mentioned protein classification problem, achieving an accuracy of 85.02%. The model includes a quadratic penalty function to decrease the chances of overfitting.

Keywords: categorical variables; numerical variables; mappings

MSC: 97-04

Citation: Alfonso Perez, G.; Castillo, R. Categorical Variable Mapping Considerations in Classification Problems: Protein Application. *Mathematics* **2023**, *11*, 279. https://doi.org/10.3390/math11020279

Academic Editors: Hongying Liu and Fanhua Shang

Received: 24 November 2022
Revised: 15 December 2022
Accepted: 4 January 2023
Published: 5 January 2023

1. Introduction

Machine learning applications have been successful in different classification tasks in areas such as physics [1–3], chemistry [4–9], and engineering [10–12], and many different algorithms currently exist, such as Trees [13–15], K-Nearest Neighbors (KNNs) [16–18], or Support Vector Machines (SVMs) [19–21]. The internal logic of these machine learning algorithms can substantially vary among the different types of models. A machine learning approach might be advantageous in a situation in which more traditional models do not exist or when these models are too complex to be efficiently implemented. Typically, machine learning models do not require a detailed understanding of the underlying problem that they are trying to model (requiring only some input and output data) or when such detailed modeling is too costly from a computational (or economic) point of view. Therefore, machine learning techniques might be suitable for modeling some complex processes [22–25] such as protein classification. In this article, we focused on the classification task of small proteins and numerical simulations regarding some assumptions regarding the mapping of categorical values, which is an issue directly related to protein modeling, as the input is typically a chain of amino acids, with each amino acid designated with a given letter. A frequently mentioned drawback of this type of approach is that machine learning techniques tend to be black boxes [26–28]. In other words, even if the classification estimations are accurate, the underlying logic is not easily explainable. In this type of modeling, some

categorical variables are commonly mapped into numerical values, as it is frequently more convenient to use numerical data in the simulations [29,30]. In this paper, we present a series of four mapping assumptions in the context of protein classification [31,32]. There is a relatively high degree of arbitrariness in the way in which these categorical variables are mapped into numerical values, and it appears interesting to test a series of assumptions about these mapping with numerical simulations. This, in fact, is one of the motivations of this article, as the issue of categorical mapping in the context of protein classification could have some modeling implications.

An algorithm is also presented for the optimization of artificial neural networks [33–35] for the classification problem, including a penalty function [36,37]. The objective of the penalty function is to favor simpler models among classification models that have similar precision. Simpler models, for example, neural networks with fewer neurons, have the advantage of being less prone to overfitting [38,39]. Overfitting is a relatively common issue in which the selected model fits rather accurately to the training data but is not properly generalized when faced with new data. Optimization approaches are commonly used in many diverse fields, such as machine learning applications in the context of ambient music in gyms [40]. There are many different proteins and many different classifications of proteins, thus making this type of analysis challenging, which is a potential limitation of the analysis. In order to minimize this risk, a well-known database of proteins (Protein Data Bank (PDB)) was selected, and we used its standard classification of proteins.

This paper is structured into five sections. An introduction in which some of the basic concepts and the main theme of the article are presented, a literature review in which related works are reported, a materials and methods section in which the four mathematical assumptions about the categorical mapping are stated, as well as the optimization algorithm, data, and general procedures. The last two sections are the results and the conclusions and recommendations, in which we analyze the results, propose potential areas of future research, and suggest some recommendations in this type of analysis.

2. Literature Review

The field of protein modeling using machine learning techniques is rapidly expanding [41]. For instance, Xu et al. used machine learning techniques to describe sequence/activity relationship [42]. This article also focused on mutations, which is an area out of the scope of our analysis. Another interesting article in the field is that of Salau and Jain [43]. In this article, the authors used machine learning techniques for the classification of cell decisions for AKT proteins. The authors used, among other techniques, neural networks such as radial basis functions (RBFs) and multi-layer perceptron (MLP). The importance of feature extraction in this context is frequently mentioned in the literature [44], and it is not exclusive to neural networks, as other popular machine learning techniques such as KNN and SVMs are also mentioned in the literature [45]. There are also some articles such as Hancock et al. [46], highlighting the importance of categorical information in machine learning techniques. More precisely, this article is a survey of categorical information in neural network applications. Some authors, such as Ofer et al. [47], followed a different approach by using natural language processing (NLP) for this type of protein classification task, which avoids the issue of categorical classification (mapping from a categorical value to a numerical value). This, however, remains an approach not followed by the majority of researchers. A potential reason for this is that a numerical approach facilitates the application of some well-known machine learning techniques, and there is so far no indication that this type of NLP approach can generate more accurate results than more traditional machine learning approaches.

Many authors, such as McDowall and Hunter [48] and Nanni et al. [49], revealed the complexity of manually performing protein classification, which is probably one of the reasons for the increasing number of applications of machine learning techniques in this field. Diplaris et al. [50] explicitly mentioned the need for automated tools that can classify new proteins. Data availability has also increased [31]. Using SVMs, Cai et al. managed to achieve an accuracy ranging from 69.1% to 99.6% [51]. Another related field

in protein classification that has raised interest from a machine learning approach is the field of protein–protein interaction. In this type of research, the objective is not to identify the type of protein but to forecast the protein–protein interaction. To some degree, the classification of the type of protein could have some impact on the interaction, but it is likely not the determining factor. Bock et al. [52] achieved an 80% success rate in this type of protein–protein interaction analysis. More recently, Das and Chakrabarti [53] followed a similar approach, achieving comparable results. For the special case of G-protein-coupled receptors, Karchin et al. [54] also followed a machine learning approach, using several techniques and obtaining an error ranging from 13.7% to 49%.

In this article, we focus on the analysis of small proteins, which is an area of increasing medical interest [55–57]. We also focus on the analysis of the classification of categorical variables, i.e., amino acids (in different representations), without the need to use NLP approaches. As previously mentioned, the importance of the process of categorical mapping into numerical values has been frequently mentioned in the existing literature. There are some articles using machine learning applications in the field of small proteins. For example, Ernest et al. [58] used this approach to study antimicrobial peptides. This research in antimicrobial peptides is actually one of the subfields that have received more interest among researchers [59,60], but there is some existing research in other areas as well, for example, regarding antifungal peptides [61].

3. Materials and Methods

Mapping variables is a common practice in machine learning applications such as classification problems [62], particularly in situations in which it is necessary to model a process using categorical variables, for example, a protein classification task using their amino acid chains. There is a certain degree of arbitrariness in this process. A protein P can be described by its amino acid chain. This can be seen with an example, as illustrated in Equation (1).

$$P = \{AC...A\} \tag{1}$$

where each amino acid is defined with its standard letter. Note that the letter B is not typically associated with an amino acid. It is usually more convenient in machine learning applications to map into numerical values. A common practice is to map it using alphabetical order and increasing numbers (Equation (2)).

$$\{A, C, ...\} \rightarrow f_1\{A, C, ...\} = \{1, 2, ...\} \tag{2}$$

As previously mentioned, this type of mapping is a bit arbitrary, as other numbers could have been used. For example, this should be equivalent to a mapping function that is identical to the previous, but a constant α is added to all the values.

$$f_2\{A, C, ...\} = \{1 + \alpha, 2 + \alpha, ...\} \tag{3}$$

This could be noted as (Equation (4))

$$f_1 \leftrightarrow f_2 \tag{4}$$

Assumption 1 (translation). *A mapping function (Equation (5))*

$$f_1\{C_1, C_2, ...\} = \{a_1, a_2, ...\} \tag{5}$$

where $\{C_1, C_2, ...\}$ *are categorical values, and* $\{a_1, a_2, ...\}$ *are numerical values, should be equivalent* $(f_1 \leftrightarrow f_2)$ *to a mapping function* f_2 *such that (Equation (6))*

$$f_2\{C_1, C_2, ...\} = \{a_1 + \alpha, a_2 + \alpha, ...\} \tag{6}$$

with $\alpha \geq 0$ *as a constant.*

Assumption 1 could be understood as a translation of mapping with a constant α. Similarly, there is no reason in principle to assume that the numerical values shown in Equation (2) are specifically representative of the related amino acid; hence, a permutation of these values (assigned to each amino acid) should generate another equivalent mapping. For example, mapping according to Equation (7)

$$\{A, C, ...\} \rightarrow f_3\{A, C, ...\} = \{2, 1, ...\} \tag{7}$$

should be equivalent to f_1 ($f_1 \leftrightarrow f_3$). In this example, the amino acids A and C are mapped into the values 1 and 2, respectively, in mapping f_1, and to the values 2 and 1 in mapping f_3. This change should have no effect on the output of a machine learning classification analysis.

Assumption 2 (permutation)**.** *A mapping function (Equation (8))*

$$f_1\{C_1, C_2, ...\} = \{a_1, a_2, ...\} \tag{8}$$

with $\{C_1, C_2, ...\}$ *as categorical values and* $\{a_1, a_2, ...\}$ *as numerical values should be equivalent to mapping according to* f_3*, as described in Equation (9).*

$$f_3\{C_1, C_2, ...\} = \{a_1, a_2, ..., a_j, a_{j-1}, ...\} \tag{9}$$

where we have a permutation of the numerical values of f_1 *in* f_3*.*

Another common situation in machine learning classification analysis is having data vectors of different lengths, for example, a group of proteins with different numbers of amino acids. These types of data are frequently stored in a matrix for easy use. It is more practical to use a square matrix, and hence a common practice is to add additional zeros (or other numerical values) to the amino acid chains to make them all of the same dimensions. We can define an operator $L()$ such that (Equation (10))

$$L(P^i) = L(\{a_1, a_2, ...\}) = l \tag{10}$$

where P^i is a given protein, and l is the length of the vector (number of amino acids) in this protein. Given a set of k proteins, the maximum size ($\bar{l}$) can be defined as Equation (11):

$$\bar{l} = sup(L(P^1), ..., L(P^k)) \tag{11}$$

Hence, $\forall P^i,\ L(P^i) \leq \bar{l}$. The set of these proteins can be represented as Equation (12):

$$X = (P^1, P^2, ...P^l) = \begin{pmatrix} a_1^1 & a_1^2 & \cdots & a_1^k \\ a_2^1 & a_2^2 & \cdots & a_2^k \\ \vdots & \vdots & \vdots & \vdots \\ \beta & \beta & \cdots & a_{\bar{l}}^k \end{pmatrix} \tag{12}$$

where β is a constant (usually set equal to zero or to a positive value) added in order to make the dimensions of the data vector containing each protein the same. Through this process, we ordered the proteins for clarity purposes (Equation (13)), but this is not a requirement.

$$L(P^1) \leq L(P^1) \leq \cdots \leq L(P^k) \tag{13}$$

As previously mentioned, the constant ($\beta \geq 0$) added to the data is arbitrary, and hence it should not impact the output of a machine learning classification estimation.

Assumption 3 (constant). *A mapping function (Equation (14))*

$$f_1\{C_1, C_2, ...\} = \{a_1, a_2, ..., \beta_1\} \tag{14}$$

where $\beta_1 \geq 0$ is an added constant to fit the required dimensions is equivalent to a mapping function f_4 (Equation (15)).

$$f_4\{C_1, C_2, ...\} = \{a_1, a_2, ..., \beta_2\} \tag{15}$$

$\forall\ \beta_2 \geq \beta_1\ \geq 0$.

3.1. Comparable Mappings

In Assumptions 1–3, the mappings are presumed to be equivalents. A less strict requirement (comparable mapping) can be also assumed on similar (but not strictly equivalent) mapping representations. For example, a protein can be described by the number of each type of amino acid and some other indicators such as the length of the chain. In this case, the mapping function would be (Equation (16)):

$$f_5\{C_1, C_2, ...\} = \{na_1, na_2, ..., na_{20}, ...\} \tag{16}$$

where na_i is the number of amino acids of type a_i contained in the chain. It should be noted that the information contained in this mapping is less than in f_1, as it is typically assumed that the order in which the amino acids appear is an important factor in determining the shape and function of the protein [63–65]. Therefore, it cannot be claimed that f_1 and f_5 are equivalent. We denote this as a comparable (but not equivalent) mapping, as expressed in Equation (17).

$$f_1 \backsim f_5 \tag{17}$$

Note that in f_5, some additional terms, such as the length of the chain, are not explicitly shown for simplicity. A potential full depiction of f_5 could be (Equation (18)).

$$f_5\{C_1, C_2, ...\} = \{l^*, na_1, na_2, ..., na_{20}, \overline{M}, \underline{M}, (\overline{M} - \underline{M}), l^*(\overline{M} - \underline{M})\} \tag{18}$$

with the terms l^*, $\overline{M}$, and $\underline{M}$ defined in Equations (19)–(21).

$$l^* = card\{C_1, C_2, ...\} \tag{19}$$

$$\overline{M} = sup\{na_1, na_2, ..., na_{20}\} \tag{20}$$

$$\underline{M} = inf\{na_1, na_2, ..., na_{20}\} \tag{21}$$

with this mapping, the information for each protein is represented with a vector of length 25. This information can be also represented by a $5x5$ matrix.

$$A = \begin{pmatrix} l^* & na_1 & na_2 & na_3 & na_4 \\ na_5 & na_6 & na_7 & na_8 & na_9 \\ na_{10} & na_{11} & na_{12} & na_{13} & na_{14} \\ na_{15} & na_{16} & na_{17} & na_{18} & na_{19} \\ na_{20} & \overline{M} & \underline{M} & (\overline{M} - \underline{M}) & l^*(\overline{M} - \underline{M}) \end{pmatrix} \tag{22}$$

A comparable representation (Equation (23)) would be the eigenvalues of this matrix $|A - \lambda I| = 0$.

$$f_6\{C_1, C_2, ...\} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\} \tag{23}$$

Hence, k proteins could be represented as (Equation (24)):

$$\begin{pmatrix} \lambda_1^1 & \lambda_1^2 & \cdots & \lambda_1^k \\ \lambda_2^1 & \lambda_2^2 & \cdots & \lambda_2^k \\ \lambda_3^1 & \lambda_3^2 & \cdots & \lambda_3^k \\ \lambda_4^1 & \lambda_4^2 & \cdots & \lambda_4^k \\ \lambda_5^1 & \lambda_5^2 & \cdots & \lambda_5^k \end{pmatrix} \tag{24}$$

Assumption 4 (eigenvalues). *For some applications, mappings f_1 and f_6 are comparable ($f_1 \leftrightsquigarrow f_6$).*

When using only five variables per protein, f_6 is more compact than f_1 compared with an arbitrary, large amount for f_1 (depending on the length of the amino acid chain). Assumptions 1–4 will be tested in a later section.

The eigenvalue approach could be considered a feature selection approach. Feature selection is an important component in machine learning approaches [66]. A simplified flowchart can be seen in Figure 1.

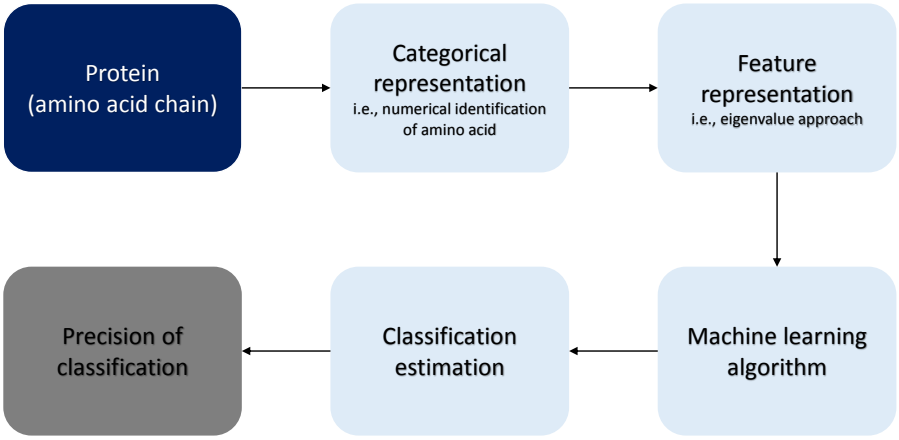

Figure 1. Simplified flowchart diagram.

3.2. Optimization

In this section, we present an algorithm for the optimization of the structure of an artificial neural network. The steps are as follows:

1. Chose the number of simulations (k), the required accuracy C_m, the maximum number of iterations (j_m), and the maximum number of neurons $\overline{a}$.
2. Define a penalty function P. For example,

$$P = \omega a^2 \tag{25}$$

   where a is the number of neurons, and ω is a constant.
3. Obtain a randomly generated number of neurons (a), with $1 \leq a \leq \overline{a} \in I$.
4. Store a classification vector $Y = \{y_1, y_2, ...\}$ (target vector) with $y_i = \{0, 1\}$ and the mapping into a matrix X.
5. Divide the data into a training dataset $\{X_T, Y_T\}$ and a testing dataset $\{X_E, Y_E\}$ [67–69].
6. Train the network (ϕ) with the training dataset $(\phi(X_T, Y_T))$.
7. Estimate the classification estimations $(Y_T^F = Y_T^F(\phi(X_T, Y_T)))$.
8. Estimate b_i as follows:

$$If \begin{cases} Y_{T,i}^F = Y_{T,i} \Rightarrow b_i = 0 \\ Y_{T,i}^F \neq Y_{T,i} \Rightarrow b_i = 1 \end{cases} \tag{26}$$

9. Estimate the accuracy Ac (Equation (27)) and calculate the additional metrics of precision (Pr), recall (Rec), and F1-score (F1) using Equations (28)–(30), respectively.

$$Ac = \frac{\sum b_i}{i} \tag{27}$$

$$Pr = \frac{TP}{TP + FP} \tag{28}$$

$$Rec = \frac{TP}{TP + FN} \tag{29}$$

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{30}$$

In this notation, TP, FP, and FN are the true-positive, false-positive, and false-negative values, respectively.

10. The estimated adjusted accuracy Ac^* is expressed in Equation (31):

$$Ac^* = Ac - P(a) \tag{31}$$

This term penalizes an overly complex model with too many neurons.

11. Compare the results of iterations and choose the model.
    - $j = 0 \Rightarrow MN(j) = Ac^*$
    - $j \neq 0$

$$If \begin{cases} Ac^* > MN(j-1) \Rightarrow MN(j) = Ac^* \\ Ac^* \leq MN(j-1) \Rightarrow MN(j) = MN(j-1) \end{cases} \tag{32}$$

12. Iterate until $(j = j_m)$ or $MN(j) \geq C_m$.
13. Repeat k times generating $MN = \{MN^1, MN^2, ..., MN^k\}$.
14. Select $\overline{MN} = sup\{MN^1, MN^2, .., MN^k\}$.
15. Calculate the classification estimations (Equation (33)) with the testing dataset for the mode $\overline{MN}$.

$$Y_E^F = Y_E^F(\phi(x_E, Y_E)) \tag{33}$$

16. Repeat step 7 with Y_E^F to obtain the testing dataset accuracy.

3.3. Data

A total of 307 small proteins were analyzed using their amino acid sequence. The data were obtained from the Protein Data Bank (PDB) [70–72]. This database is a frequently used database for protein information [73–77]. For the numerical simulations, we used the protein classification used in PDB. All the analyzed molecules were either classified as asymmetric or cyclic. A categorical variable was assigned to these two types of proteins. The dataset was composed of 254 asymmetric and 53 symmetric small proteins. The median and average number of amino acids were 84 and 81, respectively, and the amino acid chain ranged from 26 to 225 amino acids.

$$Y = Y\{0,1\} = \begin{pmatrix} Asymmetric \\ Cyclic \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{34}$$

The full list of the analyzed molecules can be seen in the Supplementary Material file. All the results shown were estimated using only the testing dataset. In machine learning, it is often not difficult to create a model that accurately describes the training dataset but fails to generalize when faced with new (unseen) data. The training dataset contains approximately 66.6% of the proteins, and the testing dataset contains the remainder 33.3%. Examples of cyclic (Figure 2) and asymmetric (Figure 3) are shown below.

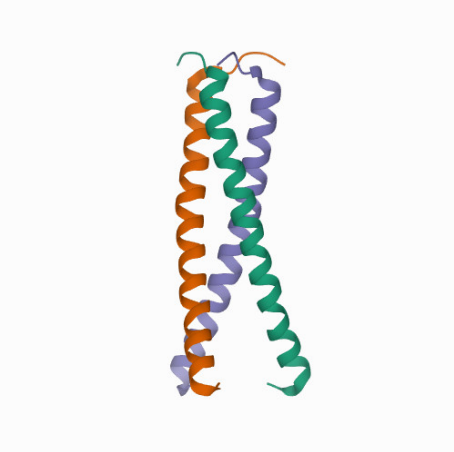

Figure 2. Cyclic molecule example (PKD2L1, Polycystin-L). Extracted from the Protein Data Bank (4GIF).

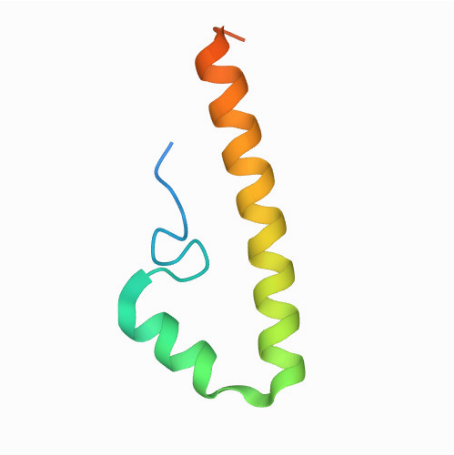

Figure 3. Asymmetric molecule example (S. cerevisiae Rtf1). Extracted from the Protein Data Bank (5EMX).

3.4. Numerical Simulations

Numerical simulations were carried out to test Assumptions 1 to 4. There could be a sizeable difference in accuracy in classification results when using different machine learning algorithms. In order to account for this, a relatively large number (23) of classification algorithms were used. The list of the algorithms used in this study can be found in Table 1. Each model was simulated q times in order to obtain a mean value for accuracy.

Table 1. List of classification algorithms (and related Matlab libraries).

N.	Algorithm	N.	Algorithm
1	Complex Tree (fitctree)	13	Fine KNN (fitcknn)
2	Medium Tree (fitctree)	14	Medium KNN (fitcknn)
3	Simple Tree (fitctree)	15	Coarse KNN (fitcknn)
4	Linear Discriminant (fitcdiscr)	16	Cosine KNN (fitcknn)
5	Quadratic Discriminant (fitcdiscr)	17	Cubic KNN (fitcknn)
6	Logistic Regression (fitglm)	18	Weighted KNN (fitcknn)
7	Linear SVM (fitcsvm)	19	Boosted Trees (fitctree)
8	Quadratic SVM (fitcsvm)	20	Bagged Tress (fitctree)
9	Cubic SVM (fitcsvm)	21	Subspace Discriminant (fitcdiscr)
10	Fine Gaussian SVM (fitcsvm)	22	Subspace KNN (fitcknn)
11	Medium Gaussian SVM (fitcsvm)	23	RUSBoosted Trees (fitctree)
12	Coarse Gaussian SVM (fitcsvm)		

The optimization algorithm was applied to neural networks. The training algorithm selected was the scaled conjugate gradient with the number of neurons selected in an automated way using the optimization algorithm. The optimization algorithm was run for one million iterations, with a constant ω in the penalty function equal to 0.0001.

4. Results

4.1. Assumption 1

In addition to the base case, 4 different models, each with 23 algorithms, were used to test Assumption 1 The difference between these four models resides in the value of the translation constant α ranging from 1000 to 1,000,000 ($\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\} = 0$, 1000, 10,000, 100,000, 1,000,000). Model 1 was the base case with $\alpha = 0$. The results showing the accuracy can be seen in Figure 4, while the results showing the precision, recall, and F1-score can be seen in Appendix A in Figures A1–A3. A Kolmogorov–Smirnov test [78] was carried out comparing the base model (Model 1 with $\alpha = 0$) with the other models for each of the 23 algorithms (see Table A1 in Appendix A). The test shows that, for the majority of models and algorithms, it cannot be concluded that there is a statistically significant difference between these distributions (accuracy value).

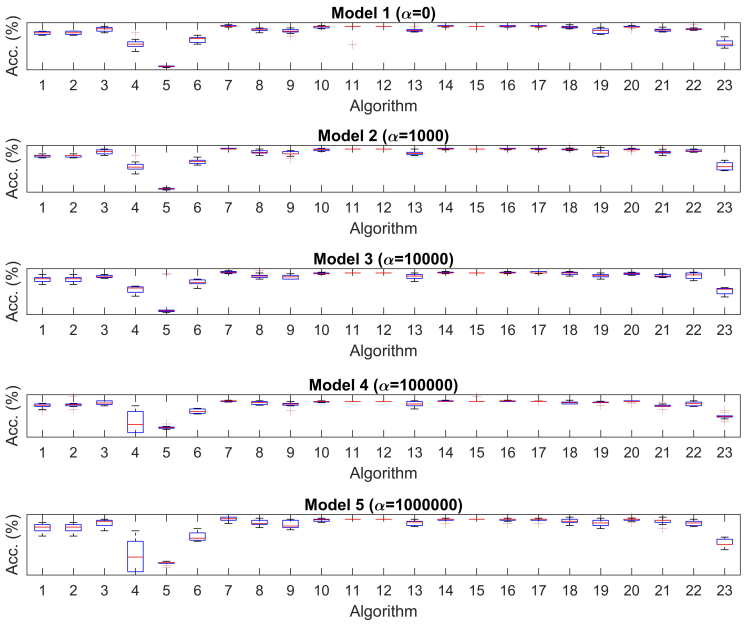

Figure 4. Numerical simulation Assumption 1. Accuracy of models after increasing the translation constant α for all the 23 algorithms. The 23 algorithms are represented in the x-axis, and the accuracy is shown in the y-axis.

4.2. Assumption 2

Five models with different permutations of the numerical values were created for all twenty-three algorithms. The number of permutations for each model was selected randomly. No additional restrictions were introduced in the permutations. The results showing the accuracy can be seen in Figure 5, while the results showing the precision, recall, and F1-score can be seen in Appendix A in Figures A4–A6. As in the previous assumption, the results for the majority of cases suggest no statistically significant difference among the majority of models and algorithms. This was also the result when using a Kolmogorov–Smirnov test comparing Model 1 with Models 6 to 9 (see Table A2 in Appendix A).

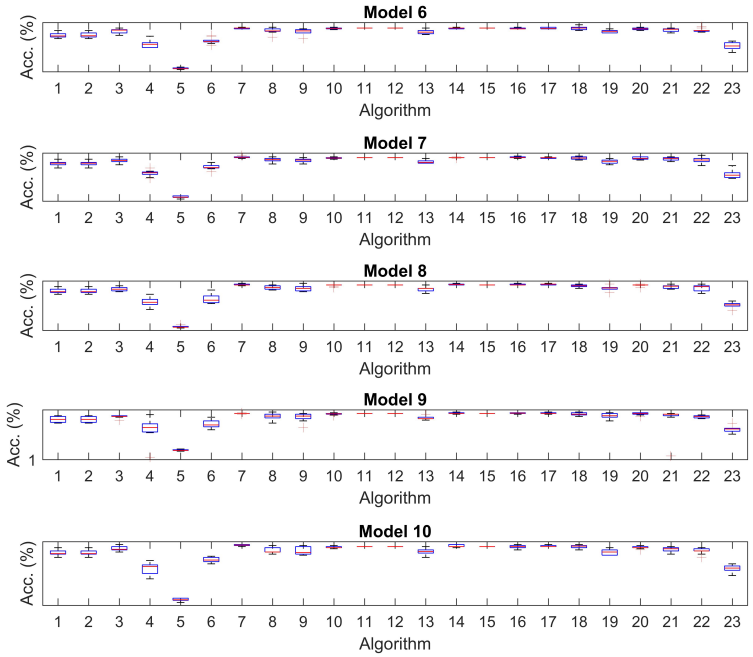

Figure 5. Numerical simulation Assumption 2. Accuracy of various models after permutations in the numerical values of the mapping. The 23 algorithms are represented in the x-axis, and the accuracy is shown in the y-axis.

4.3. Assumption 3

In this section, a variable β was added to each vector to make their length equal. The base case continued to be Model 1 with a $\beta = 0$. Four models were tested with four different betas ($\{\beta_1, \beta_2, \beta_3, \beta_4\} = \{1000, 10000, 100{,}000, 1{,}000{,}000\}$). It is worth noting that, in this case, the constant β was added in order to make the dimensions equal, and hence the existing data were not altered as in the case of Assumption 1, in which all data increased by a certain amount α. Figure 6 shows the results (accuracy) of the numerical simulations, indicating, as in the previous cases, that for most of the models and algorithms, there are no statistically significant differences. The results showing the precision, recall, and F1-Score can be seen in Appendix A in Figures A7–A9. Kolmogorov–Smirnov test comparing Model 1 with Models 11 to 14 (see Table A3 in Appendix A) generated similar results.

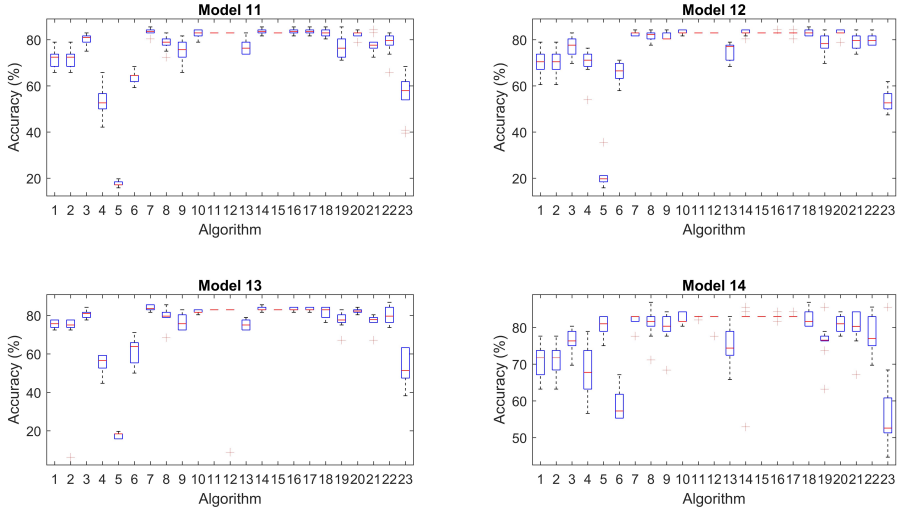

Figure 6. Numerical simulation Assumption 3 (Constant). Accuracy of various models after permutations in the numerical values of the mapping. The 23 algorithms are represented in the x-axis, and the accuracy is shown in the y-axis.

4.4. Assumption 4

In Model 15, rather than the full sequence of amino acids, the input for the classification models was the number of times that a given amino acid appeared in the amino acid chain. Hence, the information about the order of the amino acids was lost. The length of the amino acid chain was also included $(\{l^*, na_1, na_2, \ldots na_{20}\})$. Model 16 was similar to Model 15 but without the length (l^*) of the protein $(\{na_1, na_2, \ldots na_{20}\})$, see the results (accuracy) in Figures 7 and 8. The results for the precision, recall, and F1-score are shown in Appendix A in Figures A10–A15. The results of the Kolmogorov–Smirnov tests, comparing the base model (Model 1) with Models 15 and 16, showed that for the majority of the algorithms, there is no statistically significant difference, as shown in Table A4 in Appendix A.

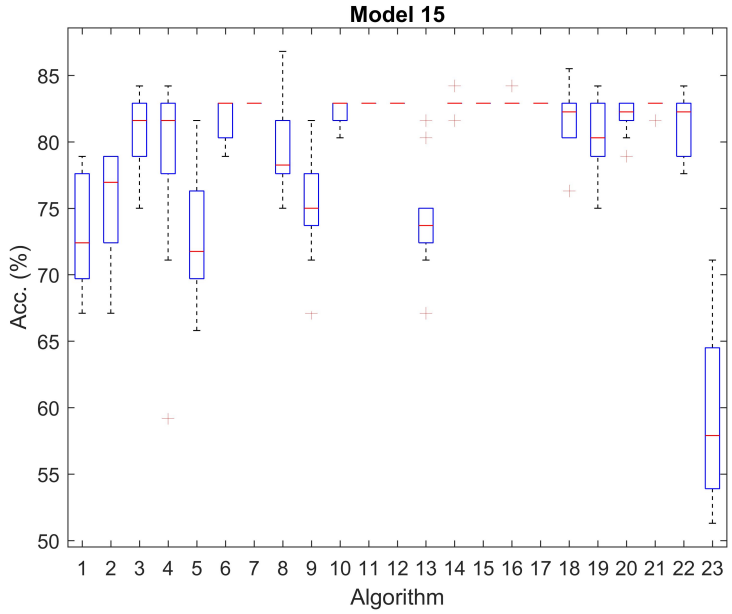

Figure 7. Accuracy of Model 15 for the different algorithms. The 23 algorithms are represented in the x-axis, and the accuracy is shown in the y-axis.

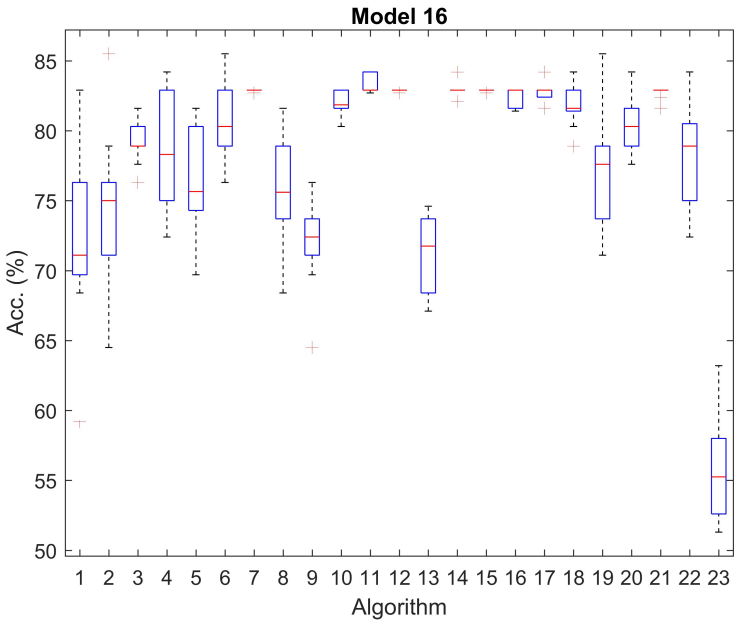

Figure 8. Accuracy of Model 16 for the different algorithms. The 23 algorithms are represented in the x-axis, and the accuracy is shown in the y-axis.

The next step entailed using the eigenvalues and some additional terms such as the l^*, as defined in expression (22). In many numerical simulations, SVM failed to generate an estimation, and they were thus excluded from the analysis. Interestingly, the rest of the models were accurate or led to better results than the base case (Model 1). The only exception to this trend was the case of the linear discriminant, in which the eigenvalue approach was statistically significantly less accurate. For all the other cases, there was no statistically significant difference, or the mean accuracy of the eigenvalue approach was higher than the value obtained in the base case. Most of the models achieved a mean accuracy above 80%. In Table 2, the mean accuracy values are shown for the eigenvalue approach and Model 1, as well as the p-values for the Kolmogorov–Smirnov test comparing these two approaches. In this table, it can be seen that the best model is the Weighted KNN model, with a mean accuracy of 83.25%, closely followed by the Subspace KNN, Simple Tree, Medium Tree, Complex Tree, and Logistic Regression, with an accuracy of 82.59%, 82.75%, 82.83%, 82.89%, and 82.95% respectively.

Table 2. Mean values of the accuracy for the eigenvalue approach and Model 1 as well as *p*-values of the Kolmogorov–Smirnov test comparing these two approaches. T refers to the average computational time required to train the algorithm.

Algorithm	Eig. Acc.	M1 Acc.	*p*-Value (ks)	T (s)
Complex Tree	82.89	72.04	0.0001	1.34
Medium Tree	82.83	72.13	0.0001	1.79
Simple Tree	82.75	78.11	0.0002	0.60
Linear Discriminant	17.08	54.44	0.0001	1.70
Quadratic Discriminant	30.37	17.47	0.6751	1.82
Logistic regression	82.95	61.87	0.0001	4.32
Fine KNN	79.88	76.53	0.0002	7.96
Medium KNN	81.59	83.18	0.6751	7.85
Coarse KNN	80.65	82.88	1.0000	7.69
Cosine KNN	81.83	82.95	0.6751	8.52
Cubic KNN	81.75	83.14	0.6751	7.10
Weighted KNN	83.25	81.83	0.0069	6.90
Boosted Trees	80.65	75.64	0.0001	7.64
Bagged Trees	82.83	81.6	0.0069	9.71
Subspace Discriminant	81.91	77.03	0.0001	10.98
Subspace KNN	82.59	78.91	0.0002	11.84
RUSBoosted Trees	81.55	54.74	0.0001	13.32

4.5. Optimization

We also used an algorithm for the optimization of the classification using neural networks, as described in Section 3.2. The algorithm was the scale conjugate gradient, and the process involved one million iterations. This model achieved an 85.02% out-of-sample classification accuracy with 215 neurons, suggesting that model parameter optimization plays an important role in improving classification accuracy. In the context of protein classification, it is important to carry out parameter optimization in a consistent way to improve the chances of the model to generalize (classify new data) with a reasonable level of accuracy. Randomly selecting the parameter could potentially lead to biases in the model or poor generalization. Figure 9 shows that the classification accuracy improves as the number of iterations increase, initially very rapidly and then more slowly as the model approaches its upper limit. There are several potential ways of performing data validation [79]. In this article, we performed cross-validation of the data in the training dataset 10 times, and then the results were tested with the testing dataset (not used during the training phase).

A limitation in this article, and a potential area of future work, is increasing the number of analyzed proteins. In this article, we analyzed 307 proteins for classification purposes, but this number could be further increased. This type of analysis could also be parallelized,

which could enable a larger number of simulations to be performed while potentially not substantially increasing computational time.

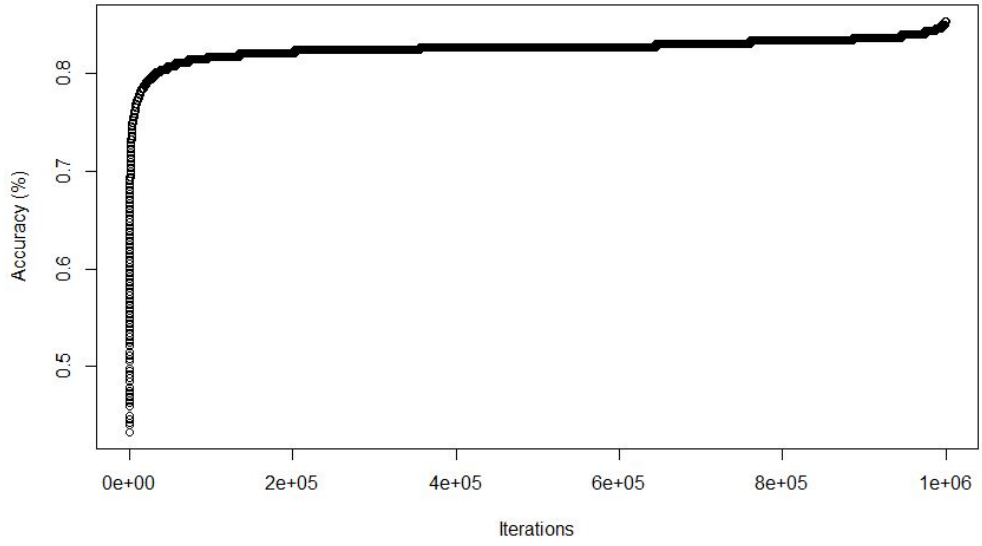

Figure 9. Improvement in accuracy as the number of iterations increases in the optimization algorithm.

5. Conclusions and Recommendations

Mapping categorical variables into numerical variables is a common practice in many machine learning classification tasks, and it is frequently carried out in an arbitrary matter. In this paper, we proposed four different assumptions related to this topic in the context of protein classification: (1) translation, (2) permutation, (3) constant, and (4) eigenvalues. Assumptions 1–3 are related to the concept of equivalent mappings in which changes to the mapping should, in principle, not alter the results of a classification analysis (for instance, adding a constant to all the input parameters). Assumption 4 relates to a less strict requirement in which the mappings are not in principle strictly equivalent, but they are comparable. An example is the eigenvalue mapping approach in which the information about the order of the amino acids (present in the initial mapping f_1) is not contained in this new mapping (f_6). The results for Assumptions 1–3 showed that, in the majority of the cases, no statistically significant difference exists between the mappings when we compared their mean accuracy. The case of Assumption 4 is different, and we see that using the eigenvalue approach generates similar or more accurate classifications than the base case model. All these numerical simulations were carried out for 23 different classification algorithms, including KNN, Tress, and SVMs. As previously mentioned, the eigenvalue approach (related to Assumption 4) generated accurate estimations for most algorithms. One noticeable exception was SVM, which, in many cases, failed to generate a classification estimation and was, therefore, excluded from the analysis. For the majority of the other algorithms, the eigenvalue approach generated results that were not statistically significantly different from the base case or that had higher mean accuracy than the base case. The best model obtained a mean classification accuracy of 83.25%. While direct comparisons are challenging, this result is 14.15% better than the lower-bound result obtained by Cai et al. [51] but lower than the upper bound. This is consistent with the idea of focusing the analysis on the stability of results rather than only focusing on increasing accuracy. This result is also substantially higher than the lower bound achieved by Karchin et al. [54], in which the authors focused on a specific subset of proteins (G-protein-coupled receptors).

An optimization analysis algorithm was also presented for the automated selection of the number of neurons in a classification model using only the frequency of the occurrence of amino acid in the amino acid chain as input (no order information), as well as the length

of the chain. The model included a quadratic penalty function to try to decrease the chance of overfitting. This approach generated an accuracy of 85.02% percent. This result is even closer to the upper bound (and substantially higher than the lower bound) of Cai et al. [51] even after accounting for the penalty function introduced to avoid an overly complex model, which potentially could impact the generalization capabilities of the model, i.e., the accuracy of the classification when faced with new data. Furthermore, this approach does not require the use of techniques such as NLP [47], which could be beneficial from an implementation point of view, as there is a large number of machine learning applications that can be easily and accurately applied to numerical values, and there is no indication that an NLP approach will generate more accurate results.

It should be noted that this accuracy is not directly comparable with the accuracy obtained in the previous sections, as there was no additional algorithm optimization. The focus of the previous section was on the comparability of the models, and hence it did not appear appropriate to add additional optimization techniques that differ in the different algorithms. For instance, an optimization process based on finding an appropriate number of neurons, as shown in the optimization section, cannot be performed for other classification techniques such as KNN, SVM, or Trees, as they do not use artificial neurons.

This type of big data analysis is challenging and can be computationally expensive, depending on the type of machine learning applied and/or the optimization algorithm followed. As an area of future research, it would be interesting to use genetic algorithms or particle algorithms as potential optimization strategies. There is a wide range of options to optimize this type of analysis. There is, however, the risk of overfitting the model, and some measures should be taken to minimize that risk, such as using a penalty function, as we used in this article, to penalize the accuracy of overly complex models. Arguably, an overly complex model is more likely to result in an overfitting issue than a simpler model.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/math11020279/s1.

Author Contributions: Conceptualization, G.A.P. and R.C.; methodology, G.A.P. and R.C.; software, G.A.P.; validation, G.A.P. and R.C.; formal analysis, G.A.P. and R.C.; investigation, G.A.P. and R.C.; resources, G.A.P. and R.C.; data curation, G.A.P. and R.C.; writing—original draft preparation, G.A.P.; writing—review and editing, G.A.P. and R.C.; visualization, G.A.P. and R.C.; supervision, G.A.P. and R.C.; project administration, G.A.P. and R.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Spanish Ministerio de Ciencia, Innovación y Universidades (PGC2018-094852-B-C21), and Universitat Jaume I (UJI-B2019-43).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All the data used in this article can be accessed from the Protein Data Bank https://www.rcsb.org/ accessed on 14 December 2022.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. *p*-values of the Kolmogorov–Smirnov test Assumption 1 (translation).

M1–M2	M1–M3	M1–M4	M1–M5
0.97479	0.97479	0.31285	0.67508
0.97479	0.97479	0.031047	0.67508
1.00000	0.67508	0.67508	0.97479

Table A1. *Cont.*

M1–M2	M1–M3	M1–M4	M1–M5
1.00000	0.67508	0.11084	0.11084
1.00000	0.31285	1.89×10^{-5}	1.89×10^{-5}
0.97479	0.31285	0.97479	0.67508
1.00000	0.67508	0.31285	0.97479
0.97479	0.97479	0.11084	0.31285
0.97479	0.97479	0.67508	0.67508
1.00000	0.97479	0.97479	1.00000
0.97479	0.97479	0.97479	0.97479
1.00000	1.00000	1.00000	1.00000
1.00000	0.31285	0.67508	0.11084
1.00000	0.97479	1.00000	0.67508
1.00000	1.00000	1.00000	1.00000
1.00000	0.97479	0.67508	0.97479
1.00000	0.67508	0.97479	0.97479
0.97479	0.67508	0.67508	0.67508
1.00000	0.67508	0.031047	0.67508
1.00000	0.97479	0.11084	0.31285
1.00000	0.97479	0.11084	0.03105
0.67508	0.67508	0.31285	0.67508
0.97479	0.67508	0.11084	0.67508

Table A2. *p*-values of the Kolmogorov–Smirnov test Assumption 2 (permutation).

M1–M6	M1–M7	M1–M8	M1–M9	M1–M10
0.67508	0.67508	0.67508	0.67508	0.31285
0.97479	0.67508	0.67508	0.67508	0.31285
1.00000	0.97479	0.67508	0.31285	0.31285
0.31285	0.11084	0.67508	0.31285	0.31285
1.00000	0.67508	1.00000	1.00000	0.67508
0.67508	0.031047	0.97479	0.67508	0.31285
0.67508	0.97479	1.00000	0.97479	0.11084
0.31285	0.67508	0.67508	0.67508	0.67508
0.31285	0.11084	0.31285	0.31285	0.67508
1.00000	0.97479	0.67508	1.00000	0.97479
0.97479	0.97479	0.97479	0.97479	0.97479
1.00000	1.00000	1.00000	1.00000	1.00000
0.97479	0.97479	0.67508	0.31285	0.97479
0.67508	0.67508	0.97479	1.00000	0.67508
1.00000	1.00000	1.00000	1.00000	1.00000
0.31285	1.00000	0.67508	0.97479	0.97479
0.31285	0.97479	0.67508	0.97479	0.97479
0.67508	0.97479	1.00000	0.97479	0.67508
0.11084	0.0068986	0.11084	0.031047	0.31285
1.00000	0.67508	0.31285	0.31285	0.97479
1.00000	1.00000	0.31285	0.67508	0.67508

Table A3. *p*-values of the Kolmogorov–Smirnov test Assumption 3 (constant).

M1–M11	M1–M12	M1–M13	M1–M14
1.00000	0.67508	0.11084	0.67508
1.00000	0.67508	0.31285	0.97479
0.67508	0.67508	0.31285	0.67508
1.00000	0.0012162	0.67508	0.0068986
0.97479	0.0068986	0.97479	1.89×10^{-5}
0.67508	0.31285	0.97479	0.31285
0.97479	0.67508	0.97479	0.31285
0.67508	0.0068986	0.11084	0.031047
0.97479	0.00017012	0.67508	0.0068986
0.97479	0.31285	0.67508	0.97479
0.97479	0.97479	0.97479	1.00000
1.00000	1.00000	1.00000	1.00000
0.97479	0.67508	0.67508	0.31285
0.97479	1.00000	1.00000	0.97479
1.00000	1.00000	1.00000	1.00000
0.97479	0.67508	0.97479	0.97479
0.97479	0.97479	1.00000	0.97479
0.67508	0.31285	0.67508	1.00000
1.00000	0.67508	0.67508	0.97479
0.31285	0.11084	0.97479	0.67508
0.97479	0.31285	0.97479	0.031047
0.67508	0.67508	0.67508	0.31285
0.31285	0.67508	0.67508	0.67508

Table A4. *p*-values of the Kolmogorov–Smirnov test Models 15 and 16.

M1–M15	M1–M16
0.31285	0.67508
0.031047	0.67508
0.11084	0.31285
0.0012162	0.00017012
1.89×10^{-5}	1.89×10^{-5}
1.89×10^{-5}	1.89×10^{-5}
0.31285	0.31285
0.67508	0.31285
0.97479	0.11084
0.97479	0.97479
0.97479	0.31285
1.00000	1.00000
0.31285	0.0012162
0.67508	0.67508
1.00000	1.00000
0.67508	0.67508
0.67508	0.97479
0.97479	1.00000
0.031047	0.67508
0.97479	0.11084
0.00017012	0.00017012
0.11084	0.67508
0.31285	0.97479

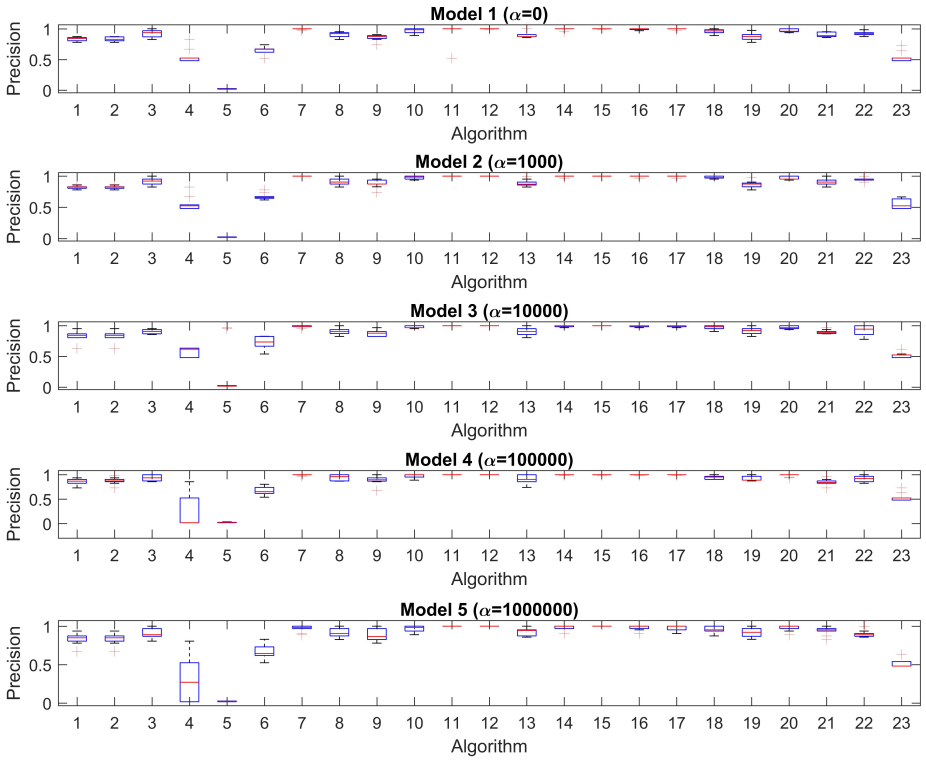

Figure A1. Numerical simulation Assumption 1. Precision of models after increasing the translation constant α for all the 23 algorithms. The 23 algorithms are represented in the x-axis, and the accuracy is shown in the y-axis.

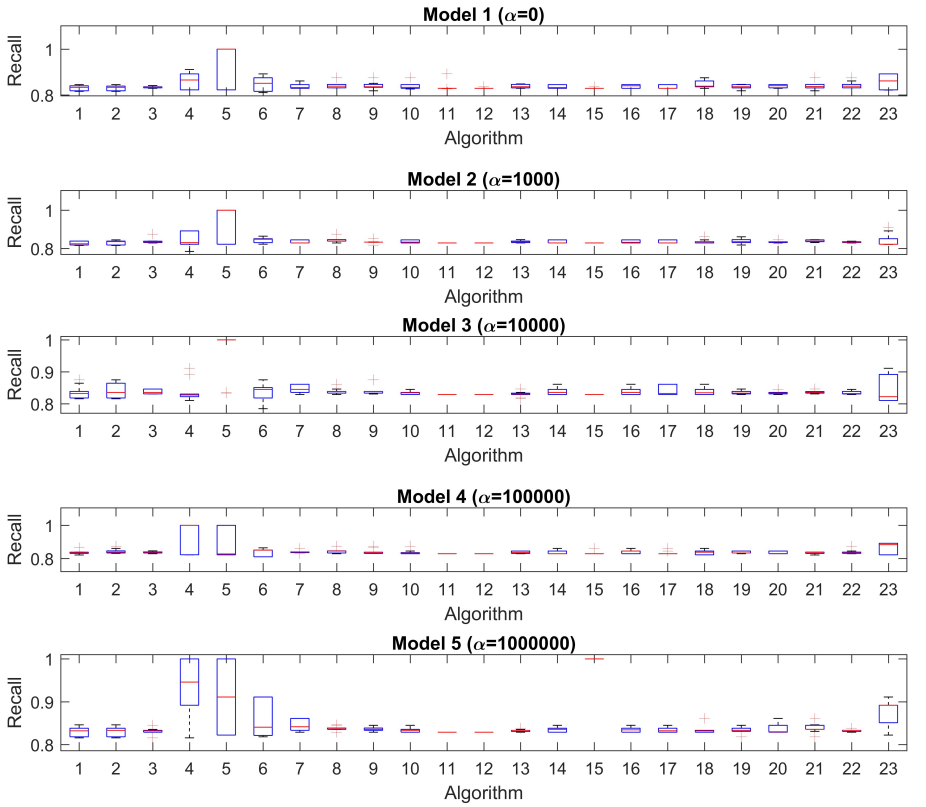

Figure A2. Numerical simulation Assumption 1. Recall of models after increasing the translation constant α for all the 23 algorithms. The 23 algorithms are represented in the x-axis, and the accuracy is shown in the y-axis.

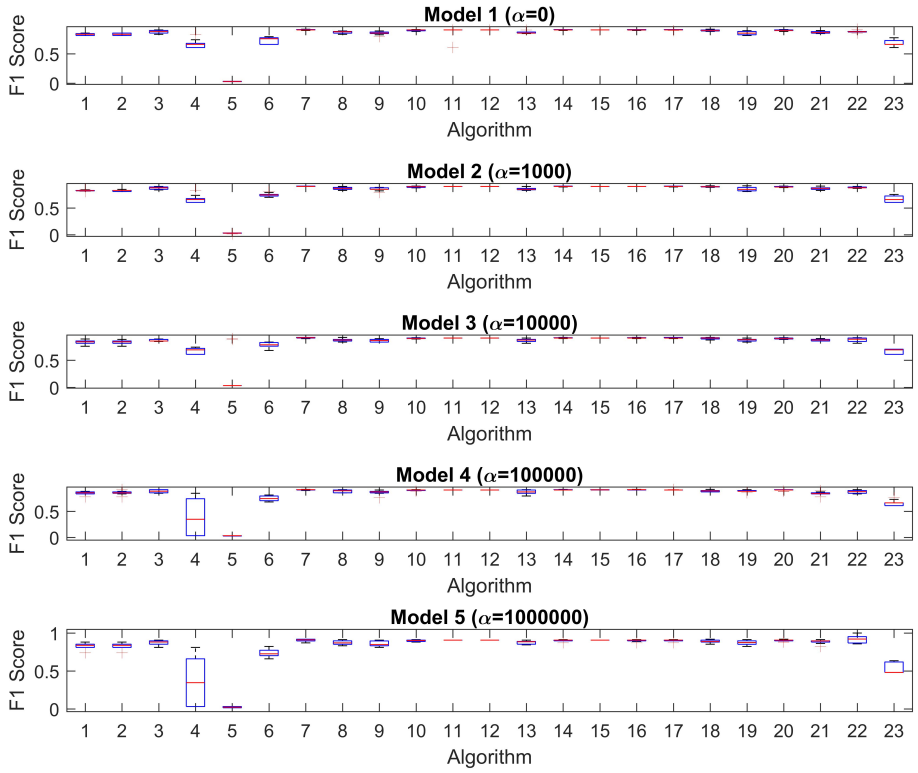

Figure A3. Numerical simulation assumption 1. F1-score of models after increasing the translation constant α for all the 23 algorithms. The 23 algorithms are represented in the x-axis, and the accuracy is shown in the y-axis.

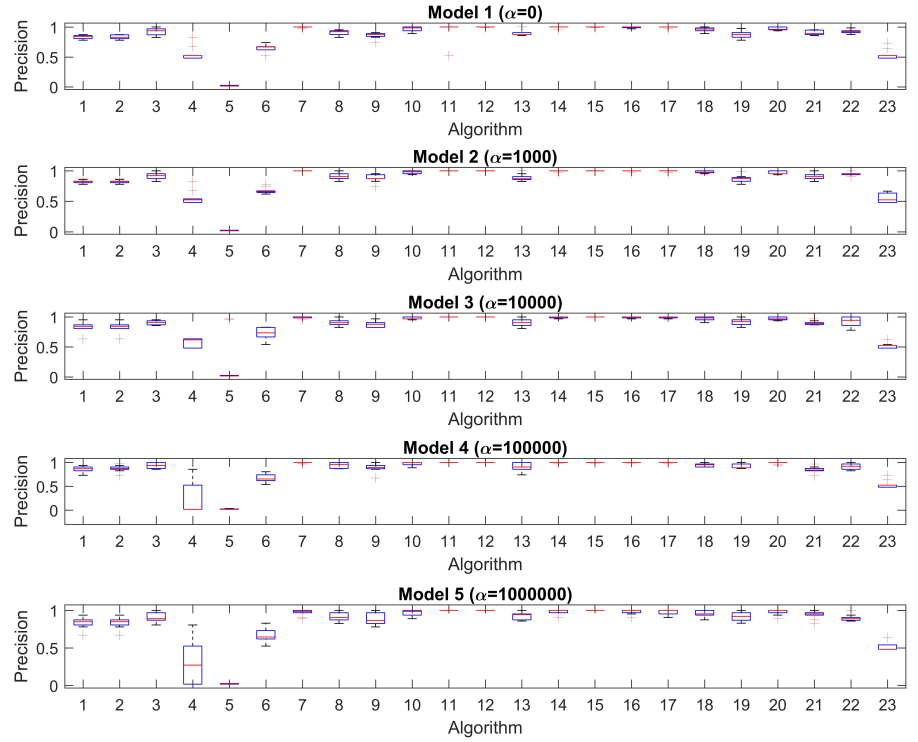

Figure A4. Numerical simulation Assumption 2. Precision of various models after permutations in the numerical values of the mapping. The 23 algorithms are represented in the x-axis, and the accuracy is shown in the y-axis.

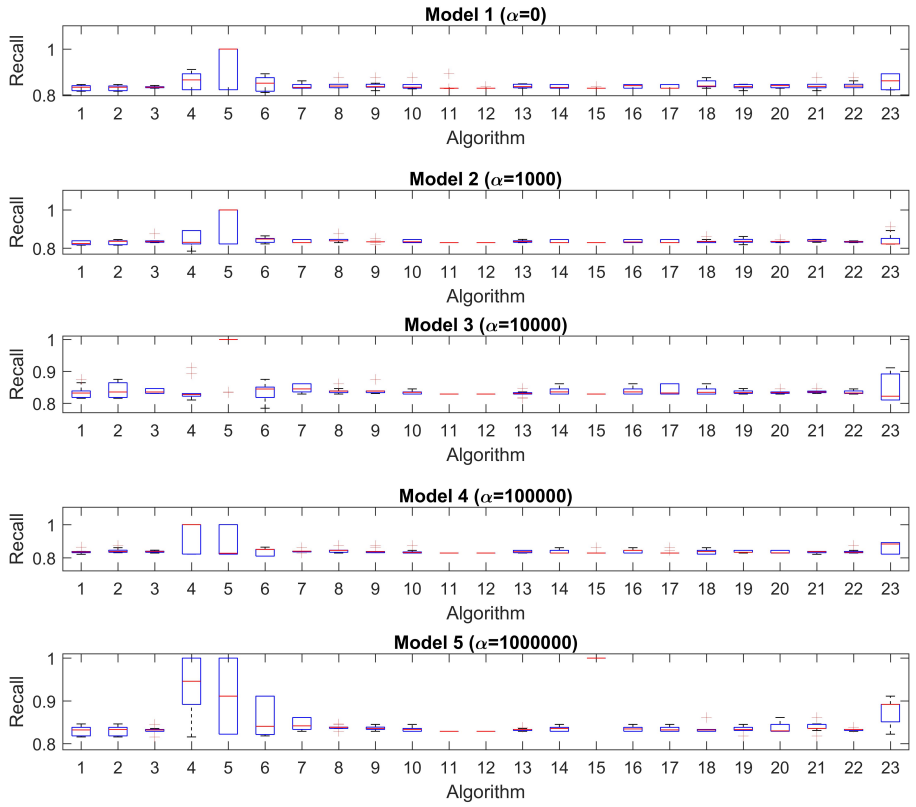

Figure A5. Numerical simulation Assumption 2. Recall of various models after permutations in the numerical values of the mapping. The 23 algorithms are represented in the x-axis, and the accuracy is shown in the y-axis.

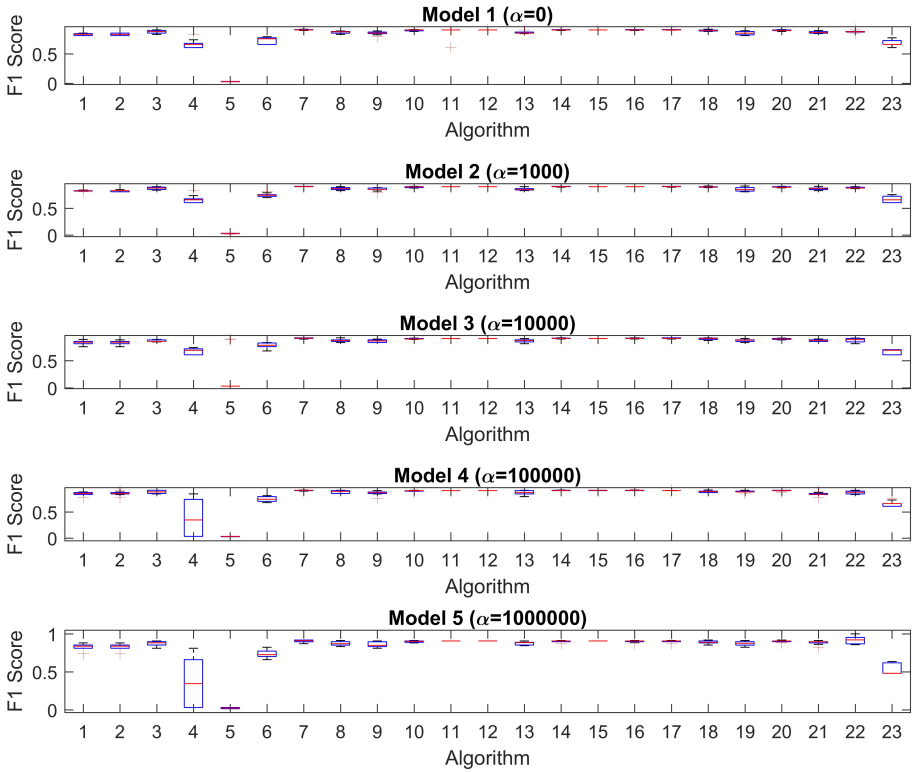

Figure A6. Numerical simulation Assumption 2. F1-score of various models after permutations in the numerical values of the mapping. The 23 algorithms are represented in the x-axis, and the accuracy is shown in the y-axis.

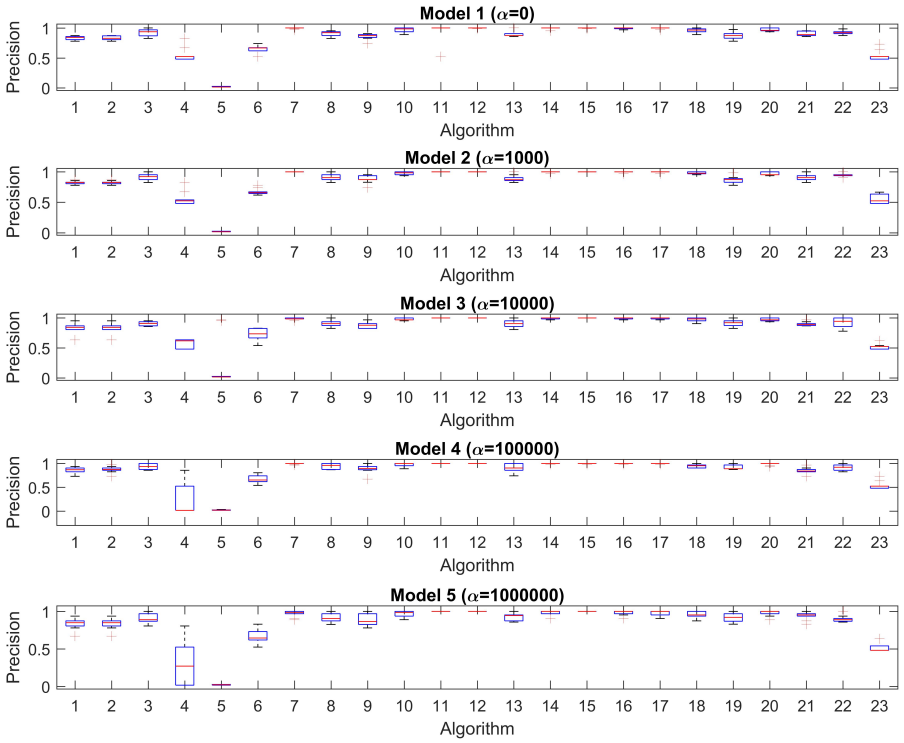

Figure A7. Numerical simulation Assumption 3 (constant). Precision of various models after permutations in the numerical values of the mapping. The 23 algorithms are represented in the x-axis, and the accuracy is shown in the y-axis.

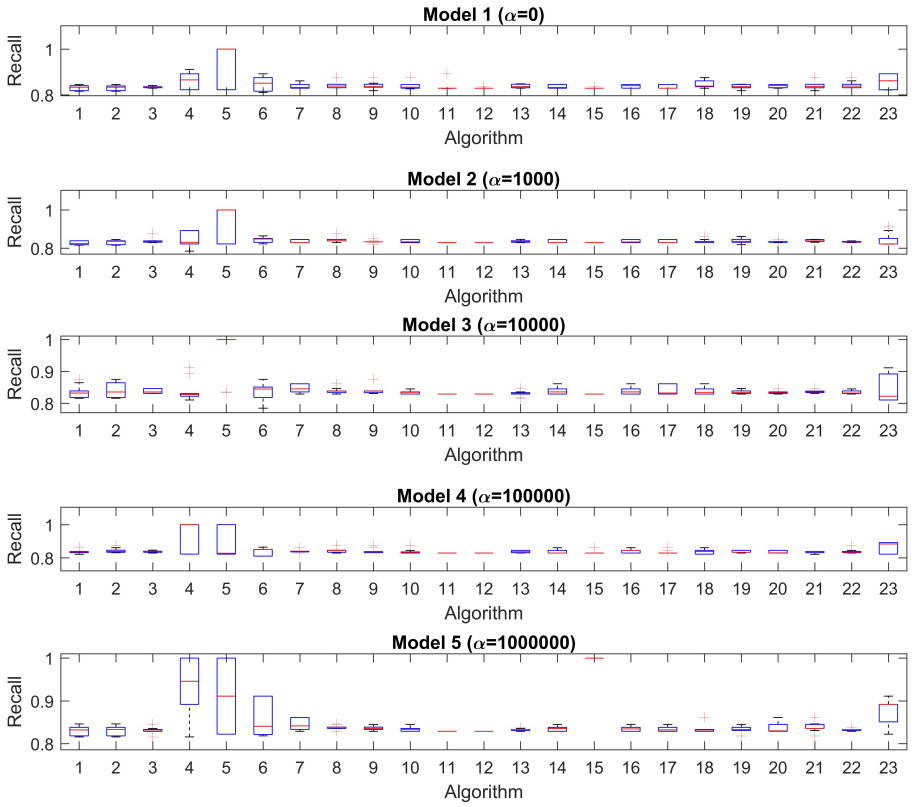

Figure A8. Numerical simulation assumption 3 (Constant). Recall of various models after permutations in the numerical values of the mapping. The 23 algorithms are represented in the x-axis, and the accuracy is shown in the y-axis.

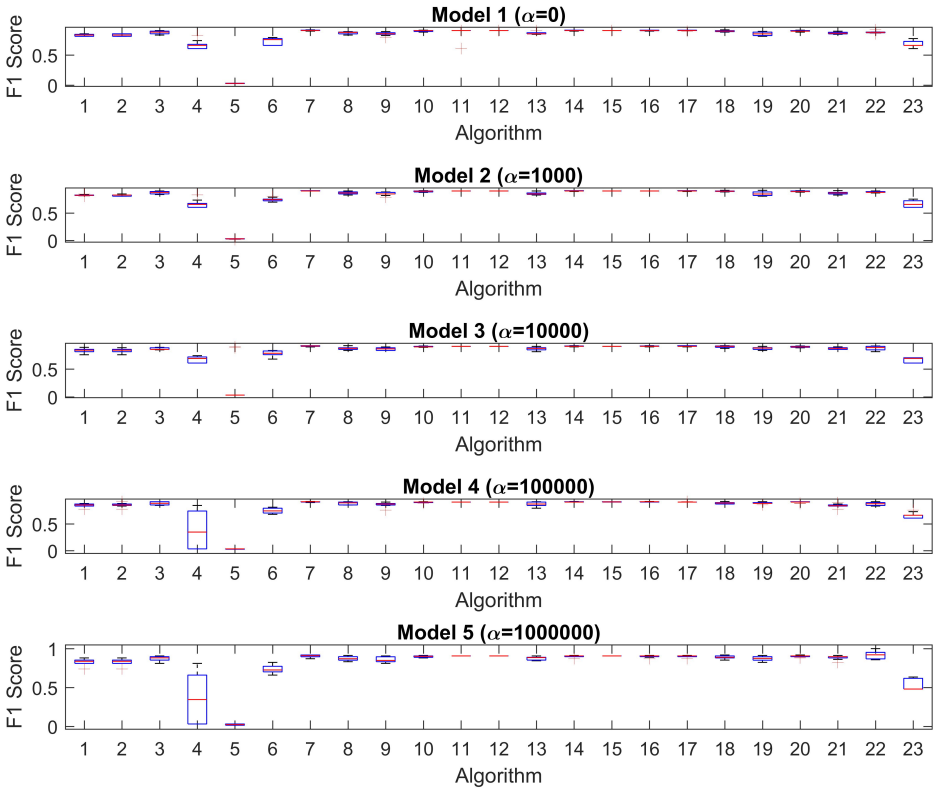

Figure A9. Numerical simulation Assumption 3 (constant). F1-score of various models after permutations in the numerical values of the mapping. The 23 algorithms are represented in the *x*-axis, and the accuracy is shown in the *y*-axis.

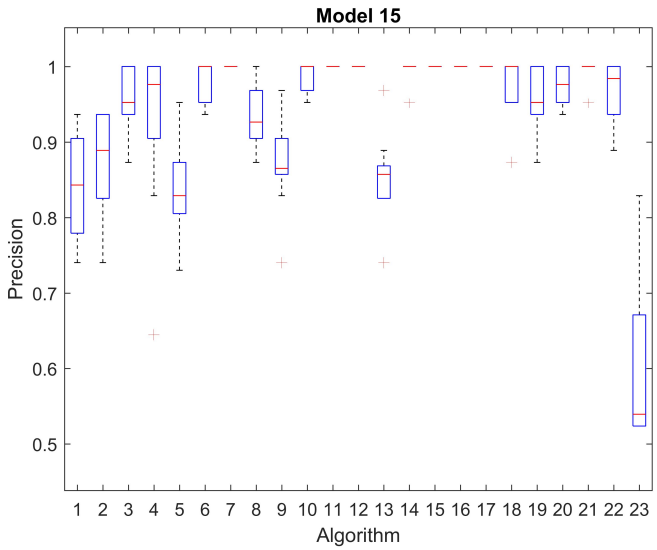

Figure A10. Precision of Model 15 for the different algorithms. The 23 algorithms are represented in the *x*-axis, and the accuracy is shown in the *y*-axis.

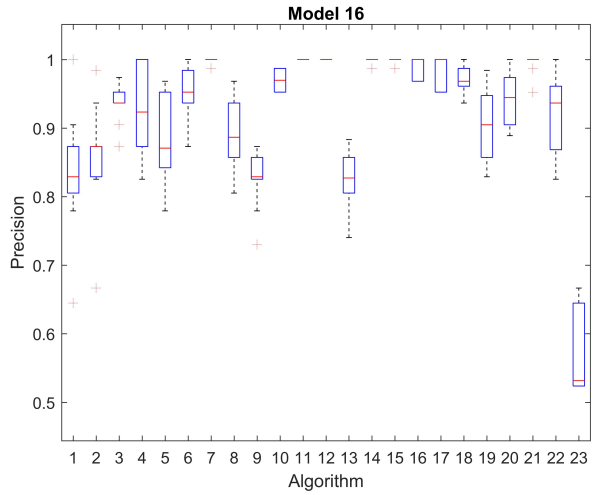

Figure A11. Precision of Model 16 for the different algorithms. The 23 algorithms are represented in the *x*-axis, and the accuracy is shown in the *y*-axis.

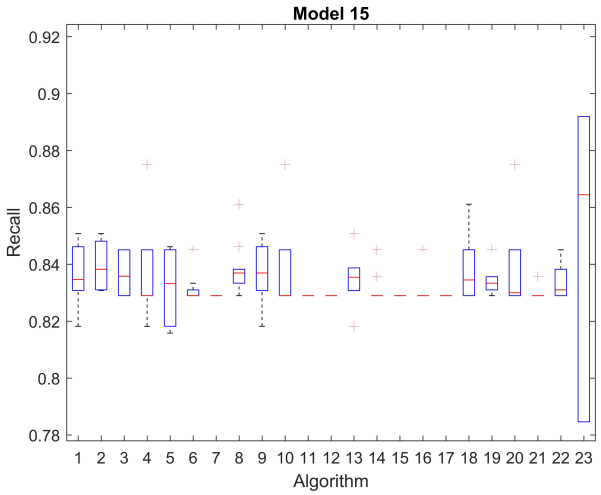

Figure A12. Recall of Model 15 for the different algorithms. The 23 algorithms are represented in the *x*-axis, and the accuracy is shown in the *y*-axis.

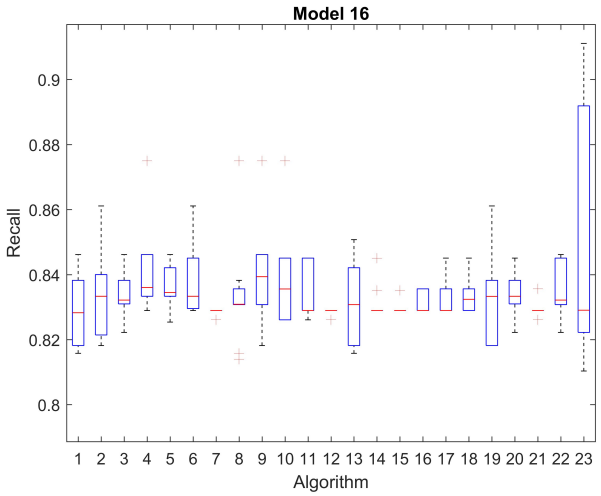

Figure A13. Recall of Model 16 for the different algorithms. The 23 algorithms are represented in the *x*-axis, and the accuracy is shown in the *y*-axis.

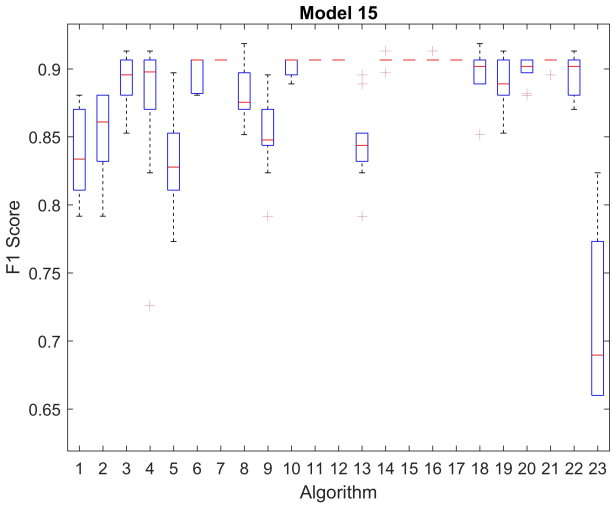

Figure A14. F1-score of Model 15 for the different algorithms. The 23 algorithms are represented in the *x*-axis, and the accuracy is shown in the *y*-axis.

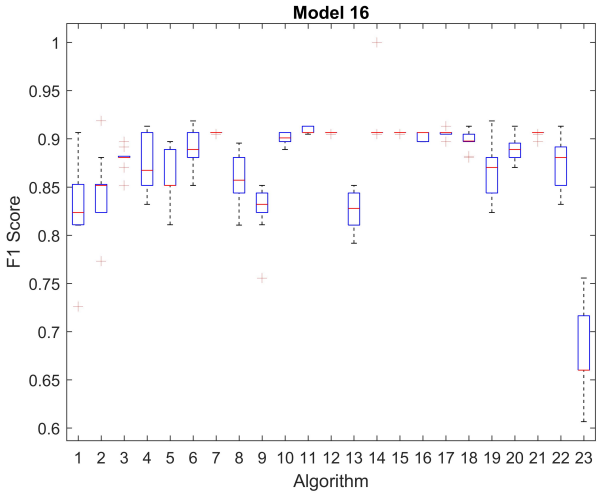

Figure A15. F1-score of Model 16 for the different algorithms. The 23 algorithms are represented in the *x*-axis, and the accuracy is shown in the *y*-axis.

References

1. Carleo, G.; Cirac, I.; Cranmer, K.; Daudet, L.; Schuld, M.; Tishby, N.; Vogt-Maranto, L.; Zdeborová, L. Machine learning and the physical sciences. *Rev. Mod. Phys.* **2019**, *91*, 45002–45041 . [CrossRef]
2. Radovic, A.; Williams, M.; Rousseau, D.; Kagan, M.; Bonacorsi, D.; Himmel, A.; Aurisano, A.; Terao, K.; Wongjirad, T. Machine learning at the energy and intensity frontiers of particle physics. *Nature* **2018**, *560*, 41–48. [CrossRef] [PubMed]
3. Karniadakis, G.E.; Kevrekidis, I.G.; Lu, L.; Perdikaris, P.; Wang, S.; Yang, L. Physics-informed machine learning. *Nat. Rev. Phys.* **2021**, *3*, 422–440. [CrossRef]
4. Jimenez, J.; Doerr, S.; Martinez-Rosell, G.; Rose, A.S.; DeFabritis, G. Deepsite: Protein-binding site predictor using 3D-convolutional neural networks. *Bioinformatics* **2017**, *19*, 3036–3042. [CrossRef] [PubMed]
5. Pages, G.; Charmettant, B.; Grudinin, S. Protein model quality assessment using 3D oriented convolutional neural networks. *Bioinformatics* **2019**, *35*, 3313–3319. [CrossRef] [PubMed]
6. Wang, X.; Terashi, G.; Christoffer, C.W.; Zhu, M.; Kihara, D. Protein docking model evaluation by 3D deep convolutional neural network. *Bioinformatics* **2020**, *36*, 2113–2118. [CrossRef] [PubMed]
7. Ragoza, M.; Hochuli, J.; Idrobo, E.; Sunseri, J.; Koes, D.R. Protein-ligand scoring with convolutional neural networks. *J. Chem. Inf. Model.* **2017**, *57*, 942–957. [CrossRef]
8. Keith, J.A.; Vassilev-Galindo, V.; Cheng, B.; Chmiela, S.; Gastegger, M.; Muller, K.; Tkatchenko, A. Combining machine learning and computational chemistry for predictive insights into chemical systems. *Chem. Rev.* **2021**, *121*, 9816–9872. [CrossRef]
9. Artrith, N.; Butler, K.T.; Coudert, F.; Han, S.; Isayev, O.; Jain, A.; Walsh, A. Best practices in machine learning for chemistry. *Nat. Chem.* **2021**, *13*, 505–508. [CrossRef]

10. Amershi, S.; Begel, A.; Bird, C.; DeLine, R.; Gall, H.; Kamar, E.; Nagappan, N.; Nushi, B.; Zimmermann, T. Software engineering for machine learning: A case study. In Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), Montreal, QC, Canada, 25–31 May 2019; pp. 291–300.
11. Park, C.; Took, C.C.; Seong, J. Machine learning in biomedical engineering. *Biomed. Eng. Lett.* **2018**, *8*, 1–3. [CrossRef]
12. Zhang, D.; Tsai, J. Machine learning and software engineering. *Softw. Qual. J.* **2003**, *11*, 87–119. [CrossRef]
13. Rodriguez-Galiano, V.; Sanchez-Castillo, M.; Chica-Olmo, M.; Chica-Rivas, M. Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines. *Ore Geol. Rev.* **2015**, *71*, 804–818. [CrossRef]
14. Blanco-Justicia, A.; Domingo-Ferrer, J. Machine learning explainability through comprehensible decision trees. In Proceedings of the International Cross-Domain Conference for Machine Learning and Knowledge Extraction, Canterbury, UK, 26–29 August 2019; pp. 15–26.
15. Allen, J.P.; Snitkin, E.; Pincus, N.B.; Hauser, A.R. Forest and trees: Exploring bacterial virulence with genome-wide association studies and machine learning. *Trends Microbiol.* **2021**, *29*, 621–633. [CrossRef] [PubMed]
16. Guo, G.; Wang, H.; Bell, D.; Bi, Y.; Greer, K. *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 986–996.
17. Lee, T.; Wood, W.T.; Phrampus, B.J. A machine learning (kNN) approach to predicting global seafloor total organic carbon. *Wiley Online Libr.* **2019**, *33*, 37–46. [CrossRef]
18. Zhang, S.; Li, X.; Zong, M.; Zhu, X.; Cheng, D. Learning k for knn classification. *ACM Trans. Intell. Syst. Technol.* **2017**, *8*, 1–19. [CrossRef]
19. Noble, W.S. What is a support vector machine? *Nat. Biol.* **2006**, *24*, 1565–1567. [CrossRef] [PubMed]
20. Cortes, C.; Vapnik, V. Support vector machine. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
21. Pisner, D.A.; Schnyer, D.M. Support vector machine. In *Chapter 6—Machine Learning*; Academic Press: Cambridge, MA, USA, 2020; pp. 101–121.
22. Qi, D.; Majda, A.J. Using machine learning to predict extreme events in complex systems. *Proc. Natl. Acad. Sci. USA* **2020**, *117*, 52–59. [CrossRef]
23. Qi, D.; Majda, A.J. Introduction to Focus Issue: When machine learning meets complex systems: Networks, chaos, and nonlinear dynamics. *Chaos Interdiscip. J. Nonlinear Sci.* **2020**, *30*, 063151.
24. Wood, D. A transparent open-box learning network provides insight to complex systems and a performance benchmark for more-opaque machine learning algorithms. *Adv. Geo-Energy Res.* **2018**, *2*, 148–162. [CrossRef]
25. Qin, J.; Hu, F.; Liu, Y.; Witherell, P.; Wang, C.; Rosen, D.W.; Simpson, T.; Lu, Y.; Tang, Q. Research and application of machine learning for additive manufacturing. *Addit. Manuf.* **2022**, *52*, 102691. [CrossRef]
26. Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* **2019**, *1*, 206–215. [CrossRef] [PubMed]
27. McGovern, A.; Lagerquist, R.; Gagne, D.J.; Jergensen, G.E.; Elmore, K.L.; Homeyer, C.R.; Smith, T. Making the black box more transparent: Understanding the physical implications of machine learning. *Nat. Mach. Intell.* **2019**, *100*, 2175–2199. [CrossRef]
28. Zhou, Z. Learnware: On the future of machine learning. *Front. Comput. Sci.* **2016**, *10*, 589–590. [CrossRef]
29. Cerda, P.; Varoquaux, G.; Kégl, B. Similarity encoding for learning with dirty categorical variables. *Mach. Learn.* **2018**, *8*, 1477–1494. [CrossRef]
30. Cerda, P.; Varoquaux, G.; Kégl, B. Encoding high-cardinality string categorical variables. *IEEE Trans. Knowl. Data Eng.* **2020**, *34*, 1164–1176. [CrossRef]
31. Sonego, P.; Pacurar, M.; Dhir, S.; Kertesz-Farkas, A.; Kocsor, A.; Gáspári, Z.; Leunissen, J.A.M.; Pongor, S. A protein classification benchmark collection for machine learning. *Nucleic Acids Res.* **2007**, *35*, 232–236. [CrossRef]
32. Jain, P.; Garibaldi, J.M.; Kirst, J.D. Supervised machine learning algorithms for protein structure classification. *Comput. Biol. Chem.* **2009**, *33*, 216–223. [CrossRef]
33. Muller, B.; Joachim, R.; Strickland, M.T. *Neural Networks an Introduction*; Springer Science & Business Media: Abingdon, UK, 1995.
34. Anderson, J.A. *An Introduction to Neural Networks*; MIT Press: Cambridge, MA, USA, 1995.
35. Miller, W.T.; Werbos, P.J.; Sutton, R.S. *Neural Networks for Control*; MIT Press: Cambridge, MA, USA, 1995.
36. Le, T.; Hoai, A.; Le, H.M.; Pham, D.T. Feature selection in machine learning: An exact penalty approach using a difference of convex function algorithm. *Mach. Learn.* **2015**, *101*, 163–186.
37. Jiang, M.; Meng, Z.; Shen, R. Partial Exactness for the Penalty Function of Biconvex Programming. *Entropy* **2021**, *23*, 132. [CrossRef]
38. Roelofs, R.; Shankar, V.; Recht, B.; Fridovich-Keil, S.; Hardt, M.; Miller, J.; Schmidt, L. A meta-analysis of overfitting in machine learning. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 1–11.
39. Peng, Y.; Nagata, M.H. An empirical overview of nonlinearity and overfitting in machine learning using COVID-19 data. *Chaos Solitons Fractals* **2020**, *139*, 110055. [CrossRef]
40. De Prisco, R.; Guarino, A.; Lettieri, N.; Malandrino, D.; Zaccagnino, R. Providing music service in ambient intelligence: Experiments with gym users. *Expert Syst. Appl.* **2021**, *177*, 114951. [CrossRef]
41. Kamerzell, T.J.; Middaugh, C.R. Prediction machines: Applied machine learning for therapeutic protein design and development. *J. Pharm. Sci.* **2021**, *110*, 665–681. [CrossRef]

42. Xu, Y.; Verma, D.; Sheridan, R.P.; Liaw, A.; Ma, J.; Marshall, N.M.; McIntosh, J.; Sherer, E.C.; Svetnik, V.; Johnston, J.M. Deep Dive into Machine Learning Models for Protein Engineering. *J. Chem. Inf. Model.* **2020**, *60*, 2773–2790. [CrossRef]
43. Salau, A.O.; Jain, S. Adaptive diagnostic machine learning technique for classification of cell decisions for AKT protein. *Inform. Med. Unlocked* **2021**, *23*, 100511. [CrossRef]
44. Salau, A.O.; Jain, S. Computational modeling and experimental analysis for the diagnosis of cell survival/death for Akt protein. *J. Genet. Eng. Biotechnol.* **2020**, *18*, 1–10. [CrossRef]
45. Jain, S.; Salau, A.O. An image feature selection approach for dimensionality reduction based on kNN and SVM for AkT proteins. *Cogent Eng.* **2019**, *6*, 1599537. [CrossRef]
46. Hancock, J.T.; Khoshgoftaar, T.M. Survey on categorical data for neural networks. *J. Big Data* **2020**, *7*, 1–41. [CrossRef]
47. Ofer, D.; Brandes, N.; Linial, M. The language of proteins: NLP, machine learning & protein sequences. *Comput. Struct. Biotechnol. J.* **2021**, *19*, 1750–1758.
48. McDowall, J.; Hunter, S. InterPro protein classification. *Bioinform. Comp. Proteom.* **2011**, *694*, 37–47.
49. Nanni, L.; Lumini, A.; Brahnam, S. An empirical study of different approaches for protein classification. *Sci. World J.* **2014**, *2014*, 236717. [CrossRef]
50. Diplaris, S.; Tsoumakas, G.; Mitkas, P.A.; Vlahavas, I. Protein classification with multiple algorithms. *Panhellenic Conf. Inform.* **2005**, *7*, 448–456.
51. Cai, C.Z.; Han, L.Y.; Ji, Z.L.; Chen, X.; Chen, Y.Z. SVM-Prot: Web-based support vector machine software for functional classification of a protein from its primary sequence. *Nucleic Acids Res.* **2003**, *31*, 3692–3697. [CrossRef]
52. Bock, J.R.; Gough, D.A. Predicting protein–protein interactions from primary structure. *Bioinformatics* **2001**, *17*, 455–460. [CrossRef]
53. Das, S.; Chakrabarti, S. Classification and prediction of protein–protein interaction interface using machine learning algorithm. *Sci. Rep.* **2021**, *11*, 1761. [CrossRef]
54. Karchin, R.; Karplus, K.; Haussler, D. Classifying G-protein coupled receptors with support vector machines. *Bioinformatics* **2002**, *18*, 147–159. [CrossRef]
55. Chen, X.; Gao, Y.; Wang, Y.; Pan, G. Mussel-inspired peptide mimicking: An emerging strategy for surface bioengineering of medical implants. *Smart Mater. Med.* **2021**, *2*, 26–37. [CrossRef]
56. Kazemzadeh-Narbat, M.; Cheng, H.; Chabok, R.; Alvarez, M.M.; De La Fuente-Nunez, C.; Phillips, K.S.; Khademhosseini, A. Strategies for antimicrobial peptide coatings on medical devices: A review and regulatory science perspective. *Crit. Rev. Biotechnol.* **2021**, *41*, 94–120. [CrossRef]
57. Apostolopoulos, V.; Bojarska, J.; Chai, T.-T.; Elnagdy, S.; Kaczmarek, K.; Matsoukas, J.; New, R.; Parang, K.; Lopez, O.P.; Parhiz, H. A global review on short peptides: Frontiers and perspectives. *Molecules* **2021**, *26*, 430. [CrossRef]
58. Charoenkwan, P.; Chiangjong, W.; Hasan, M.M.; Nantasenamat, C.; Shoombuatong, W. Review and Comparative Analysis of Machine Learning-based Predictors for Predicting and Analyzing Anti-angiogenic Peptides. *Curr. Med. Chem.* **2022**, *29*, 849–864. [CrossRef] [PubMed]
59. Fjell, C.D.; Jenssen, H.; Hilpert, K.; Cheung, W.A.; Pante, N.; Hancock, R.E.W.; Cherkasov, A. Identification of novel antibacterial peptides by chemoinformatics and machine learning. *J. Med. Chem.* **2009**, *52*, 2006–2015. [CrossRef] [PubMed]
60. Rondon-Villarreal, P.; Sierra, D.; Torres, R. Machine learning in the rational design of antimicrobial peptides. *Curr. Comput.-Aided Drug Des.* **2014**, *10*, 183–190. [CrossRef] [PubMed]
61. Mousavizadegan, M.; Mohabatkar, H. An evaluation on different machine learning algorithms for classification and prediction of antifungal peptides. *Med. Chem.* **2016**, *12*, 795–800. [CrossRef]
62. Sen, P.C.; Hajra, M.; Ghosh, M. Supervised classification algorithms in machine learning: A survey and review. *Emerg. Technol. Model. Graph.* **2020**, *937*, 99–111.
63. Ivankov, D.N.; Finkelstein, A.V. Prediction of protein folding rates from the amino acid sequence predicted secondary structure. *Proc. Natl. Acad. Sci. USA* **2004**, *101*, 8942–8944. [CrossRef]
64. Kunt, I.D.; Crippen, G.M.; Kollman, P.A. Calculation of protein tertiary structure. *J. Mol. Biol.* **1976**, *106*, 983–994.
65. Hagler, A.T.; Barry, H. On the formation of the protein tertiary structure on a computer. *Proc. Natl. Acad. Sci. USA* **1978**, *75*, 554–558. [CrossRef]
66. Salau, A.O.; Jain, S. Feature Extraction: A Survey of the Types, Techniques, Applications. In Proceedings of the 2019 International Conference on Signal Processing and Communication (ICSC), Noida, India, 7–9 March 2019; Volume 75, pp. 158–164.
67. Zur, R.M.; Jiang, Y.P.; Pesce, L.L. Noise injection for training artificial neural networks. A comparison with weight decay and early stopping. *Med. Phys.* **2009**, *36*, 4810–4818. [CrossRef]
68. Lu, H.; Setiono, R.; Huan, L. Effective data mining using neural networks. *IEE Trans. Knowl. Data Eng.* **1996**, *8*, 957–961.
69. Torgyn, S.; Khovanova, N.A. Handling limited datasets with neural networks applications: A small data approach. *Artif. Intell. Med.* **2017**, *75*, 51–63.
70. Rose, P.W.; Prlic, A.; Altunkaya, A.; Bi, C.; Bradley, A.R.; Christie, C.H. The RCSB protein data bank: Integrative view of protein, gene and 3D structural information. *Nucleic Acids Res.* **2016**, *gkw1000*, 271–281 .
71. Rose, P.W.; Bi, C.; Bluhm, W.F.; Christie, C.H.; Dimitropoulus, D.; Dutta, S.; Bourne, P.E. The RCSB Protein data bank: New resources for research and education. *Nucleic Acids Res.* **2012**, *41*, 475–482. [CrossRef]
72. Berman, H.M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T.N.; Weissig, H.; Bourne, P.E. The protein data bank. *Nucleic Acids Res.* **2000**, *28*, 235–242. [CrossRef]

73. Springs, R.V.; Artymiuk, P.J.; Willet, W. Searching for 3D patterns of amino acids in 3D protein structures. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 412–421. [CrossRef]
74. Abola, E.E.; Bernstein, F.C.; Frances, C.; Koetzle, T.F. The protein data bank. *Neutroms in Biology*; Springer: Boston, MA, USA, 1984.
75. Berman, H.M.; Henrick, K.; Nakamura, H. Announcing the worldwide protein data bank. *Nat. Struct. Mol. Biol.* **2003**, *10*, 980. [CrossRef]
76. Parasuraman, S. Protein data bank. *J. Pharmacol. Pharmacother.* **2012**, *3*, 351 [CrossRef]
77. Sussman, J.L.; Abola, E.E.; Lin, D.; Jiang, J.; Manning, N.O. The protein data bank. *Struct. Biol. Funct. Genom.* **1999**, *54*, 251–264.
78. Fauman, E.B.; Hyde, C. An optimal variant to gene distance window derived from an empirical definition of cis and trans protein QTLs. *BMC Bioinform.* **2022**, *23*, 1–11. [CrossRef]
79. Guarino, A.; Malandrino, D.; Zaccagnino, R. An automatic mechanism to provide privacy awareness and control over unwittingly dissemination of online private information. *Comput. Netw.* **2022**, *202*, 108614. [CrossRef]