UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingeniería

Typical Testor Selection Algorithm for Classification Models and its Application in Datasets

Mateo Martínez Mejía

Matemática

Trabajo de titulación presentado como requisito

para la obtención del titulo de

Matemático

May 10, 2023

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingeniería

HOJA DE CALIFICACIÓN DE TRABAJO DE FIN DE CARRERA

Mateo Martínez Mejía

Nombre del profesor, Titulo académico: Eduardo Alba, Ph.D.

May 10, 2023

© Derechos de Autor

Nombres y apellidos:	Mateo Martínez Mejía
Código:	00333890
Cédula de Identidad:	1719153189
Lugar y fecha:	May 10, 2023

ACLARACIÓN PARA LA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en `http://bit.ly/COPETheses`

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on `http://bit.ly/COPETheses`

Agradecimientos

Quisiera agradecer a mi familia, quienes han sido mi apoyo incondicional a lo largo de toda mi carrera y quienes han sido mi sostén en cada reto y cada decisión. Quisiera dar un agradecimiento especial a Eduardo Alba quien me ha guiado a lo largo de mi carrera y quien ha hecho posible que se lleve a cabo este trabajo. De igual manera, quiero dar gracias a mis amigo María del Carmen Salazar, José Ochoa, Alejandro Rueda y Ariana Soria por acompañarme en a lo largo de esre viaje, motivándome y ayudándome a superarme día a día. Por su soporte y motivación en cada momento quiero agradecer a Milena Mora, quien ha permanecido a mi lado siempre alentándome a seguir adelante. Por último, quiero dar gracias a la Universidad San Francisco de Quito por permitirme lograr este desarrollo académico y profesional, al igual que a mis maestros quienes han sido una pieza fundamental en mi crecimiento y quienes me han mostrado la belleza e importancia de la ciencia e investigación.

Resumen

La teoría de testores ha presentado buenos resultados como un método de reducción de dimensionalidad de bases de datos, siendo capaz de discriminar entre elementos de clases distintas. Sin embargo, la condición de discriminación utilizada en esta teoría, no garantiza una buena capacidad de clasificación. Por esta razón, buscamos evaluar los testores típicos haciendo uso de qué tan similares son los objetos dentro de su misma clase y qué tan diferentes son los objetos de clases diferentes. Realizando esta evaluación en tres bases de datos distintas y empleando dos modelos de clasificación, finalmente se pudo determinar la capacidad del algoritmo de selección de elegir los mejores testores típicos para este propósito.

Palabras clave: *Teoría de testores, testores típicos, capacidad de clasificación, algoritmo de selección, modelo de clasificación.*

Abstract

Testor theory has shown great results as a dataset dimensionality reduction method, being able to discriminate between elements from different classes. However, the condition for discrimination used in this theory does not guarantee a good classification capability. Therefore, we look to evaluate the typical testors, taking into account the similarities between elements from the same class and the distinctions between elements from different classes. Making this evaluation in three different dataset, and using two classification models, we ere able to finally determine the capacity of the selection algorithm to choose the best typical testors for this objective.

Keywords: *Testor theory, typical testors, classification capability, selection algorithm, classification model.*

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Nowadays, data plays a huge roll in the development of new technology and the progress of our society. Its acquisition, storage and use is of great importance to any advancement, which is why its optimization is essential. One may believe that with more data in a dataset, any classification model will obtain better results. Even though there is some truth in this affirmation, it is not the most efficient way to use this data. These datasets may contain large amounts of unnecessary information that does not contribute in any way to the classification model. This is where typical testors provide an important advancement. Typical testors allow us find the most relevant features in a dataset such that we are able to reduce the size of a dataset without affecting its ability to distinguish objects from different classes. Throughout this work we will find a new way to identify the best typical testors that fulfill this task in the finest way.

Testor Theory was developed during the 1960s, but with a different approach

to the one given in this work. Initially, this theory was established to detect faults within electrical circuits that executed boolean functions. However, it later developed into a feature selection theory for various problems applied to supervised learning algorithms [1, 2]. In this work, we will apply Testor Theory for feature selection, in other words, as a dimensionality reduction method.

One of the main problems in Testor Theory, is that it doesn't make any distinction between the typical testors that are found within a dataset. Therefore, in theory, every typical testor should have the same result when applied in a classification model. However, through testing, we have found that this is not the case. Due to this, we would like to find a new way in which we can select the typical testors that give the best results by considering their ability to distinguish objects from different classes, as well as, find similarities between object from the same class. Throughout this work, we will present the basic concepts of Testor Theory and the classification models used to evaluate the typical testors. Just as well, we will describe in detail the proposed typical testor selection algorithm that will allow us to rank them based on certain factors. Furthermore, we will train and test the classification model with these typical testors in order to determine the usefulness of the method, as well as its limitations. Lastly, we will evaluate these results and comment on the following steps to improve the algorithm.

Chapter 2

Methodology

A dataset is a group of objects from different classes that are characterized by a set of features. These features, and the values associated to each object, are what allow us to distinguish objects that belong to distinct classes. Therefore, we can analyze, which of these features or what subset of features contain enough information to be able to identify the correct class for each object in the dataset. In other words, we are looking for the most relevant features in the dataset. This idea is what leads us to the concept of typical testors.

Throughout this section, we will cover the methodology used for this work regarding the calculation of typical testors, their selection with the proposed algorithm, and the classification models employed to test the viability of the suggested solution. We will start with a brief introduction to Testor Theory and the mathematical background of each classification model. Furthermore, we will describe the datasets used for this work, as well as any necessary changes made to them.

Lastly, we will describe in detail the process taken in order to select, what we expect to be, the best typical testors in terms of classification effectiveness.

2.1 Testor Theory

To understand Testor and the concept that allows it to perform a dimensionality reduction, it is vital to introduce some important definitions.

Let U be a set of objects with n features and r classes to which they can belong to [2, 3]. It is important to mention that the r classes are disjoint, that is, if $c_i = \{u \in U : u \textit{ belongs to class } i\}$, then $c_i \cap c_j = \varnothing \ \forall \ i, j \in \{1, 2, \ldots, r\}$.

Definition 1. Let $|c_i| = p_i$, then, the number of pairs of objects with distinct classes will be

$$N = \sum_{k=1}^{r} \sum_{i=1}^{k} p_k p_i - \sum_{i=1}^{r} p_i^2 \tag{2.1}$$

Definition 2. Let $u \in c_k$, $v \in c_l$ where $k \neq l$ and let (u, v) the i-th pair out of N possible ones, then we define the dissimilarity matrix M as follows

$$M = \{m_{ij}\}_{N \times n} \quad m_{ij} \in \{1, 0\} \tag{2.2}$$

where $m_{ij} = 0$ if obects u and v are similar in the feature j, and $m_{ij} = 1$

otherwise [3].

Having obtained this matrix, we can define the idea of testor and typical testor.

Definition 3. Let J be the set of features that define the objects in U. $T \subseteq J$ is a testor if and only if $\nexists\, u, v \in U$ such that they are similar in every feature [4, 5]. In terms of M, the matrix restricted to T, that is $M_{|T}$, does not have rows with only zeros.

Definition 4. If $\nexists\, T_2 \subset T$ such that T_2 is also a testor, then we say that T is a typical testor [4, 5].

Let us consider that the matrix M will have redundant information for the calculation of typical testors, for which we look to reduce it and only keep the rows that contain the most relevant information.

Definition 5. Let f and g be rows in matrix M. We say that $f < g$ if $\forall j \in \{1, 2, \ldots, n\},\ f_j < g_j$. Furthermore, if $\nexists\, h$ row of M such that $h < f$, then f is called a basic row of M [2, 3].

Definition 6. Let M_B be the matrix that contains every basic row in M, then we say that M_B is the basic matrix of M [3].

Now, in order to relate the set of typical testors obtained from M and the same set obtained from M_B, we may use the following proposition.

Proposition 1. Let $\Psi^*(M)$ be the set of typical testors of M, then it is true that

$$\Psi^*(M) = \Psi^*(M_B) \tag{2.3}$$

In other words, the set of typical testors of M is the same for M_B [3].

Having defined the notions of testor theory, we can observe that the subset of features selected in a typical testor, can be considered as the minimum collection of features that allows us to distinguish elements from distinct classes.

This preliminary information about typical testors, allows us to introduce one of the most important testor searching algorithms, known as *Yablonski & Compatible Sets*, also known as *YYC*.

2.1.1 YYC Algorithm

The YYC algorithm [6] is a recursive algorithm that calculates typical testors. Instead of directly searching for the typical testors of the whole basic matrix, as many other methods do, it looks to find the typical testors up to the i-th row of the basic matrix in the i-th iteration. In other words, each iteration, the algorithm updates the set of typical testors when a new row is added. The way in which this method works, is by validating the compatibility between an element of the set of typical testors up to the i-th row and an element of the set of typical testors of row $i + 1$. This means that the algorithm adds, to each typical testor, a new feature, depending on the positioning of 1s in row $i + 1$, and then checks that the tipicity property is not lost.

2.2 Classification Models

For this work, we will measure the effectiveness of the method proposed through the accuracy obtained by two different classification models. For this, we will first define the mathematical background for these models. The two classification models that will be used for this work are Support Vector Machines and Artificial Neural Networks.

2.2.1 Support Vector Machines (SVM)

The Support Vector Machine model, also known as SVM, allows us to make data classifications. Just like any other classification model, a mathematical background is what enables it to classify objects from different categories. As we will see in the following section, each datasets used for this work only has two classes, which, in turn, simplifies the SVM mathematical approach that we will cover.

The main goal of this model, is to find the best hyperplane in a high dimensionality space in terms of a generalized performance metric [7]. Before setting forward any definitions about the model, we need some context about the dataset and how every feature relates to this generalized metric. First of all, we consider a dataset with l objects, defined by n features. Let $U = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$ be the set of objects in the dataset, where $\mathbf{x}_i \in \mathbb{R}^n$ y $y_i \in \{-1, 1\}$[7]. It is important to mention that the numerical value of each component of $\mathbf{x}_i$, is given by the features that define the object. In case any features are categorical, it is simple to transform the data into numerical values for each category and maintain this

definition of the objects for the model. On the other hand, y_i represents the class to which the i-th object belongs to.

Definition 9. Let $\mathbf{w}$ be an n-dimensional vector and b a scalar, then we define the following hyperplane

$$\mathbf{x} \cdot \mathbf{w} + b = 0 \tag{2.4}$$

where $\mathbf{w}$ is orthogonal to the hyperplane. This hyperplane, separated the objects from both classes [8].

We can now define two new hyperplanes parallel to the first which do not detach any elements from each class. Such hyperplanes will be the following.

$$\mathbf{x} \cdot \mathbf{w} + b = 1 \tag{2.5}$$

$$\mathbf{x} \cdot \mathbf{w} + b = -1 \tag{2.6}$$

Proposition 3. Let γ be the distance between the parallel hyperplanes, then the following equation describes it in term of $\mathbf{w}$

$$\gamma = \frac{2}{||\mathbf{w}||} \tag{2.7}$$

Hence, the objective of the method is to maximize γ, in other words, to minimize $||\mathbf{w}||$ [7].

Definition 10. The optimization problem is the minimization of $||\mathbf{w}||^2$ under the following restriction

$$y_i[(\mathbf{x}_i \cdot \mathbf{w}) + b] - 1 \geq 0 \quad \forall i \in \{1, \ldots, l\} \tag{2.8}$$

where $\mathbf{x}_i$ is the i-th object in the dataset and y_i is the object's class [7].

Furthermore, we look to solve this problem by using its lagrangian formulation.

$$L(\mathbf{w}, \boldsymbol{\alpha}, b) = \frac{\mathbf{w}^T\mathbf{w}}{2} - \sum_{i=1}^{l} \alpha_i[y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1] \tag{2.9}$$

where $\alpha_i \geq 0 \ \forall i \in \{1, \ldots, l\}$ are the Lagrange multipliers [7].

To justify the convexity of the problem, let us consider that we look to minimize a quadratic objective function under linear restrictions. Thus, we can solve this problem through an equivalent maximization problem. To find the dual problem, we need to impose the following optimality conditions.

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l} y_i \alpha_i \mathbf{x}_i = 0 \tag{2.10}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{l} y_i \alpha_i = 0 \tag{2.11}$$

From equation 2.10, we can obtain an expression for $\mathbf{w}$ and we can replace it in equation 2.9 to obtain the objective function for the dual problem [7]. Let $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_l\}$, then

$$max(F(\boldsymbol{\alpha})) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{2.12}$$

This objective function is subjected to the restriction presented in equation 2.11 and taking into account again that $\alpha_i \geq 0 \ \forall i \in \{1, \ldots, l\}$.

By solving the dual problem, we obtain the values for the Lagrange multipliers, α_i. This result lets us define the support vectors that give name to this model.

Definition 11. Let $\mathbf{x}_i,\ \mathbf{x}_j$ be two objects from distinct classes. If α_i and α_j are different from 0, then we say that $\mathbf{x}_i$ and $\mathbf{x}_j$ are support vectors that generate the hyperplanes that separate both classes [7].

Having defined the mathematical background for an SVM model, we can proceed to define the same for Neural Network models.

2.2.2 Artificial Neural Networks

The second classification model that we will use for this work are Artificial Neural Networks (ANN or NN). This model takes a different approach to the one shown for Support Vector Machines, but is also based on an optimization process.

Let $\mathbf{x}$ be an object in the dataset, the main purpose of this classification model is to find a function φ, such that $\varphi(\mathbf{x})$ is the class to which $\mathbf{x}$ belongs to [9, 10]. To begin with this brief introduction to NN, let us consider a dataset with n features, that is, each object in the dataset may be described as $\mathbf{x} = (x_1, \ldots, x_n)$.

Definition 12. Let y be the class of object $\mathbf{x}$ and $\mathbf{w} = (w_1, \ldots, w_n)$ be the weight vector that allows us to define z as follows

$$z = (w_1, \ldots, w_n) \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + b = \mathbf{w}^T\mathbf{x} + b \tag{2.13}$$

then the final output y through some function φ is

$$y = \varphi(z) \tag{2.14}$$

[10].

To ease the notation, let $w_0 = b$ and $x_0 = 1$, and redefine $\mathbf{x} = (1, x_1, \ldots, x_n)$

and $\mathbf{w} = (b, w_1, \ldots, w_n)$. It is important to mention that function $\varphi(z)$ is called an activation function.

The following diagram extracted from [9], will helps us understand the way the model works.

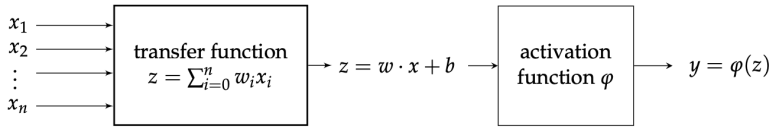

Figure 2.1: Single neuron diagram

As we can see form figure 2.1, this model is based on update the weights in order to generate the proper value for y when applying the activation function. Therefore, we look to understand the process made in order to do this. Let us consider that $\hat{y}$ is the given class, and both $y, \hat{y} \in \mathbb{Z}$.

Definition 13. Let $E(\mathbf{w}) : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a differentiable function in the neighborhood of a point $\mathbf{w} \in \mathbb{R}^n$. Then we say that E is the error function and that $-\nabla E$ is the direction to which E is fastest decreasing [9].

Therefore, we can calculate ∇E to find an expression to update the weights. Let us consider E as the square error, defined as follows.

$$E = \frac{1}{2}\sum_{i=1}^{n}(\hat{y} - y)^2 = \frac{1}{2}\sum_{i=1}^{n}(\hat{y} - \varphi(z))^2 \tag{2.15}$$

Therefore, we have the following.

$$-\nabla E(\mathbf{w}) = -\left(\frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}\right)$$

$$\implies -\frac{\partial E}{\partial w_i} = (\hat{y} - y)\varphi'(z)x_i \tag{2.16}$$

Definition 14. Let Δw_i be the value that updates the i-th component of the weight vector. Then we have the following expression

$$\Delta w_i = \eta(\hat{y} - y)\varphi'(z)x_i \tag{2.17}$$

Where $\eta \in (0, 1)$ is defined as the learning rate of the model [11].

Thus, we can update each component of the weight vector using the following recursive expression.

$$w_i = w_i + \Delta w_i \tag{2.18}$$

Therefore, we have obtained a way to update our weight vector that minimizes the error function. Hence, this learning algorithm will be able to update the weights in such a way that the computed outputs match the desired outputs and successfully classify the objects in the dataset.

2.2.3 Model Evaluation Scores

In order to evaluate how good the classification obtained from one of the previous models is, we will use two specific scores. These scores are the accuracy of the model and the AUC score. Here we will define the both of these scores in order to understand their use in the following chapter.

First of all, in order to define the accuracy of a classification model, let us define a confusion matrix since the information it contains will allow to calculate it. A confusion matrix, for a two class dataset, has the information regarding a classification model. It has four values: True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN). Both TP and TN represent the objects that where correctly classified, while FP and FN represent the objects that were incorrectly classified. Having defined this value, we have the following equation for calculating the accuracy.

$$accuracy = \frac{TP + TN}{FP + FN + TP + TN} \tag{2.19}$$

In other words, we say that the accuracy of a classification measures the ratio of correct predictions over the total number of evaluations [12].

On the other hand, in order to define the AUC score, we have to introduce the ROC curve. This curve is a representation of the sensitivity vs. 1 - specificity of a diagnostic test. Using this curve, the area under the curve (AUC) represents the effectiveness of the model. In this case, we consider that a value of 0.5 suggests

that there has been no discrimination, while a value of 1 represents a perfect classification. [13].

2.3 Datasets

Datasets showcase certain challenges when it comes to applying the concepts from Testor Theory. As we have seen, typical testor calculation requires a dissimilarity matrix which has the information about the distinction between objects from different classes. Due to this fact, the comparison method used in any features that have non-discrete values would result in a column of 1s in the dissimilarity and basic matrix. In turn, this would make the mentioned feature a typical testor by itself, which would not represent the full benefits of the method used.

Alternatively, we propose a "discretization" of any non-discrete features in the dataset. The way in which we carry out this discretization starts by analyzing the features of type *float64*. In order to maintain a similar dataset to the original one, we look to discretize each feature in the same way. In this case, we separate the each feature into ten uniform intervals (categories). Furthermore, we group the values of the feature into each of these categories being left with only ten different values and successfully discretizing the feature. This is a preliminary idea of the process to prepare the dataset for the dissimilarity and, further ahead, the typical testor calculation. Now, we will cover the process taken for each dataset.

2.3.1 Stellar Classification Dataset - SDSS17

The first dataset used for this work is the Stellar Classification Dataset - SDSS17 retrieved from the Kaggle platform [14]. As it was mentioned before, certain changes were needed in order for us to be able to use the dataset for typical testor calculation regarding the values shown in some of the features. However, before this transformation, we were also required to consider other changes to the dataset that would ease the needed calculations.

First of all, the original dataset had 3 distinct classes: "GALAXY", "STAR", and "QSO" (quasar), only the last two were used for this work. This decision was taken due to the fact that the GALAXY class has around double the amount of elements in comparison to the other two classes, which were balanced between them. In addition to this, we have to take into consideration that the dissimilarity matrix is obtained by comparing elements from different classes, and the number of comparisons are greatly increased with the addition of another class. The number of rows that this matrix has, is of great importance to the problem since the computational power is limited for the execution of the algorithms.

In a similar way, some features had to be eliminated from the original dataset since the information they contained did not provide any contrasting data between classes. Some of these features included identification information which was unique for each object which would result in the feature being a typical testor by itself, but not containing any relevant information to contrast both classes. Due to this elimination, the dataset was reduced to 11 features instead of 17. The list of features that were kept for the study is displayed in table 2.1 [14].

Feature	Description	Type
alpha	Right ascension angle	$float64$
delta	Declination angle	$float64$
u	Ultraviolet filter in the photometric system	$float64$
g	Green filter in the photometric system	$float64$
r	Red filter in the photometric system	$float64$
i	Near infrared filter in the photometric system	$float64$
z	Infrared filter in the photometric system	$float64$
field_ID	Field identification number	int
redshift	Redshift value based on the increase in wavelength	$float64$
plate	SDSS plate identification	int
fiber_ID	Identification of the fiber that pointed the light at the focal plane	int

Table 2.1: Stellar Classification Dataset Features

As it was mentioned at the beginning of the section, each of the features with a data type *float64* were discretized in order to calculate the dissimilarity matrix. It is relevant to mention that during the discretization of these features, certain considerations had to be taken into account to avoid involuntary mistakes. Note that the uniform intervals generated for the discretization were chosen by taking the maximum and minimum values present in the feature and calculating the length necessary for the intervals in order to cover the interval (min, max). There exists a particular case in which the length of these intervals could be greatly affected, which is when the maximum or minimum values of the feature are too far apart from the rest of the values. In this case, we would eliminate the element to which this value corresponds to and eliminate from the database. Following this step, we recalculate the minimum and maximum value of the feature to discretize it. This case was found only in the feature "u", and the steps aforementioned were taken to solve it.

Having finished this process we were ready to proceed to the calculation of the dissimilarity matrix, and typical testors afterwards.

2.3.2 QSAR Biodegradation Dataset

The second dataset employed for this work is the QSAR Biodegradation Dataset retrieved from the UCI Machine Learning Repository platform. This dataset contains information about 1055 chemicals that determines if they are ready biodegradable (RB), or not ready biodegradable (NRB) [15]. Just like the previous dataset, this one requires certain alterations in order to calculate the dissimilarity matrix. However, since this dataset only contains two classes it was not necessary to eliminate any other classes. Just as well, we do not require to eliminate any features and, thus, we maintain all 41. In this case, each element is characterized by 41 molecular descriptors, out of which, 17 present non-discrete values. These 17 descriptors are modified by implementing the discretization algorithm previously discussed.

Due to the high number of features in this dataset, and the fact that the database has been kept the same apart from the discretization, no table will be provided with the information of each feature. Such information may be found in [15].

Having discretized the dataset, we can proceed to calculate its dissmilarity matrix and, eventually its typical testors.

2.3.3 Credit Approval Dataset

This dataset is the Credit Approval Dataset retrieved from the UCI Machine Learning Repository platform. It contains information regarding credit card applications, although, to protect the confidentiality of the data, the features don't contain any labels [16]. With a total of 690 objects, the dataset divides into two classes, "+" with 307 objects, and "−" with 383 objects. There exists a particular issue with this dataset, which is the fact that there are some values missing, and we had to address this problem to begin working with it.

Before making any changes to the dataset, we have to take into account that any value given for this missing data will have an effect during the discretization of any features with data type $float64$. This was the case for one feature which had a total of 12 missing values. The approach taken for this case was to appropriately fill these values in such a way that they are different from the rest, but affect minimally the discretization. That is, this values are the only ones within their interval, but the rest of the values are evenly distributed within the other 9 intervals. This was achieved by giving a value of "0" to these missing spaces. For every other missing value in categorical features, we simply added a new category for these values.

As we mentioned before, the information regarding the features has been censored in order to protect the confidentiality of the data. Therefore, we are not able to provide any further information about this. Any other additional information may be found in [16].

2.4 Typical Testor Calculation

Throughout this section we will describe the approaches taken to calculate the typical testors for each of the datasets previously described. Depending on the length of the dataset, we calculated the typical testors in different ways in order to avoid lengthy calculations that are not necessary.

2.4.1 Stellar Classification Dataset - SDSS17

For this dataset, we have to consider that the number of elements is fairly large and the dissimilarity matrix would be as well. The "QSO" class has a total of 18961 objects, while the "STAR" class has a total 21543 objects. Therefore, the resulting dissimilarity matrix would be have a dimension of 408476823×11, which is exceedingly extensive, and, thus, it would take a long time to reduce it to basic matrix. Given this, we took a different approach.

Even though the way in which we addressed this situation would cause to not calculate the typical testors of the entire dataset, any calculated from a considerable subset of objects should maintain their property of distinguishing classes with any new objects. From this idea, we took the following process. First of all, we separated the objects from different classes and randomly permuted each list in order to select 3000 random objects from each one. We must take three groups of 1000 objects from each subset of elements and compare the i-th group from class "QSO" with the i-th group from class "STAR", thus we are left with three 1000000×11 dissimilarity matrices. Later, we reduce each matrix to a basic

matrix, concatenate them and reduce it again to a basic matrix.

Having obtained this final basic matrix, we can finally calculate the typical testors associated with it. This calculation is done through the YYC algorithm.

2.4.2 QSAR Biodegradation Dataset

In this case, the dataset only contained 854 elements, for which we used a straightforward method in order to calculate the basic matrix and typical testors as described in Testor Theory. Hence, we generated a dissimilarity matrix by comparing every pair of objects from different classes, in this case, ready biodegradable and not ready biodegradable. From this dissimilarity matrix, we reduce it to basic matrix and calculate the typical testors by using the YYC algorithm.

2.4.3 Credit Approval Dataset

Just like with the QSAR Biodegradation Dataset, with only 690 elements, this dataset doesn't require any new approach for its typical testor calculation. We compare the elements from the "+" and "-" classes in order to calculate the dissimilarity matrix. Afterwards, we calculate the basic matrix and its typical testors.

2.5 Typical Testor Selection Algorithm

From the previous section, we have seen that through Testor Theory we are able to calculate a set of typical testors without making any distinction between them. Therefore, we put forward an idea that helps us rank the typical testors in order to identify which ones will provide a better classification.

As we can see from Testor Theory, typical testors have the property of presenting the least number of features that allow us to differentiate between two elements of different classes. Now, the following question presents itself. Could these features present differences between elements of the same class? Following this line of thought, one can wonder how could typical testors be related to similarities between elements of the same class. Thus, we look to take into account these similarities when we choose the proper typical testors.

In order to analyze these similarities, we must first define certain notions to extend Testor Theory. Let us remember that we defined U as set of of objects with n features and r classes to which they can belong to. Thus, we have the following definitions.

Let $|c_i| = p_i$, then, the number of pairs of objects in the class c_i will be

$$N_{s_i} = \sum_{k=1}^{p_i} p_i - k = \frac{p_i(p_i - 1)}{2} \tag{2.20}$$

Now, we can define the similarity matrices SM_k for each class in the following

way.

Definition 15. Let $u \in c_k$, $v \in c_k$ and (u, v) be the i-th pair out of N_{s_k} possible ones, then we define the similarity matrix SM

$$SM_k = \{(sm)_{ij}\}_{N_{s_k} \times n} \quad (sm)_{ij} \in \{1, 0\} \tag{2.21}$$

where $(sm)_{ij} = 0$ if objects u and v are different in the feature j, and $(sm)_{ij} = 1$ otherwise.

Having defined the r similarity matrices, we put forward an idea to score the typical testors based on these matrices. Let us consider that we look to have the most similarities within each class, which is why we score each similarity matrix individually. The way in which we calculate each score is by calculating the percentage of 1s in the similarity matrix restricted to the typical testor.

Definition 16. Let SM_k be the k-th similarity matrix and TT a typical testor, then we score the TT by using the following equation

$$s_k = \frac{1}{N_{s_k} \times |TT|} \sum_{i=1}^{N_{s_k}} \sum_{j \in TT} \{(sm)_{ij}\} \tag{2.22}$$

Now, we have to use this r scores obtained in order to calculate a generalized score for the typical testor. This is done in the following way.

Definition 17. Let s_k be the score for the typical testor TT in the k-th simi-

larity matrix SM_k, then the similarity score for the typical testor is the following

$$ss = \prod_{k=1}^{r} s_k \tag{2.23}$$

Notice that we calculate the product of every score. We take this approach since we look to ensure that we have the most similarities within all of the classes. Therefore, we can rank the typical testors by taking into account the score obtained from equation 2.23. However, this approach can be improved even further.

Let us consider that the similarity score (ss) only considers 1s within each similarity matrix, but does not take into account the difference between elements from different classes. The typical testors let us find the features for which each pair of elements from distinct classes are different in at least one of them, however, they do not give us information about how different these objects are. Hence we look to take a similar approach as the one with the similarity matrices, but with the dissimilarity matrix.

Definition 18. Let M be the dissimilarity matrix, then the dissimilarity score for the typical testor TT is the following

$$ds = \frac{1}{N \times |TT|} \sum_{i=1}^{N} \sum_{j \in TT} \{m_{ij}\} \tag{2.24}$$

Now, we must find a way to use both scores in order to rank the typical testors. Let us consider that before working with both scores, we must first normalize the

similarity score with respect to the maximum value obtained for ss by a typical testor. That is, we divide each value of ss by the largest value obtained. Let us denote ss_n as the normalized similarity score for a typical testor.

Definition 19. Let ds and ss_n be the dissimilarity and normalized similarity scores respectively, then the score for typical testor TT is the following

$$score = ds + ss_n \tag{2.25}$$

Finally, we have reached a score that considers both similarities and differences between the objects in the dataset and allowing us to rank the typical testors previously obtained.

Throughout the following section, we will analyze the usefulness of this method in the different datasets, contrasting the efficiency of a classification model by using the best and worst typical testors found by applying this metric.

Chapter 3

Results

As we mentioned in the previous chapter, we look to analyze the convenience of the proposed typical testor ranking algorithm by considering the efficiency of the classification model used. Throughout this chapter, we will observe the results obtained from the typical testor calculation and evaluate the new selection algorithm based on the accuracy of the model training and AUC value obtained for each one. We look to interpret whether, or not, the best typical testors present an advantage when used to train a classification model, as opposed to the worst classified typical testors.

3.1 Typical Testor Calculation

From the previous chapter, we were able to calculate the typical testors for each dataset. The following table presents the summarized information about the datasets and their corresponding typical testors.

Dataset	No. Features	No. TT	Min. Length	Max. Length
Star Class	11	19	2	4
QSAR Biodeg	41	11533	11	19
Credit	15	6	8	9

Table 3.1: Typical testor calculation summary

As we can see from table 3.1, the number of typical testors obtained from each dataset greatly varies. Just as well, we can see that the dataset reduction proposed by the typical testors lets us eliminate around at least 40% of the original features in all three datasets, and more than 50% in two of them. This, together with the results obtained from the classification effectiveness tests done later on, show the usefulness of Testor Theory as a dimensionality reduction method.

Having shown the results obtained from the typical testor calculations, we can proceed to apply the selection algorithm and test the classification model with these typical testors.

3.2 Classification Model Scores

In order to analyze the effectiveness of each classification model, we must separate the objects from the dataset into a training set and a testing set. In order to

avoid any bias within the training of the model, the testing objects were randomly selected. For both classification models, we took 33% of the data for the testing set, the other 67% was used to train the model.

Here, we present both the accuracy score and AUC score, corresponding to each model when using the selected typical testors. Just as well, we will present these scores for the model trained with the full set of features, in order to compare the usefulness of the typical testors as a dimensionality reduction method.

It is important to mention that we will not present any other performance metric, such as the F1, Recall, True Negative Rate (TNR), True Positive Rate (TPR), False Positive Rate (FPR), False Negative Rate (FNR) and F1 score. This is due to the fact that the AUC score allows us to consider the balance between the number of elements in each class of the dataset, while the other scores don't. In a similar way, other scores only consider some of the values obtained in a confusion matrix, while the accuracy score will use every value, giving us a more general idea.

3.2.1 Star Classification Dataset (SVM Model)

No. Features	**Accuracy**	**AUC score**
11	99.54 %	0.9955

Table 3.2: Full dataset SVM training accuracy (Star Classification)

No. Features	**Accuracy**	**AUC score**
4	99.89 %	0.9990
4	99.89%	0.9990
4	99.89%	0.9990

Table 3.3: Best typical testors SVM training accuracy (Star Classification)

As we can see from tables 3.2,3.3, the top selected typical testors present a classification accuracy and AUC score even better to those calculated for the full set of features. Thus, the reduction achieved by Testor Theory was successful for the classification provided by an SVM model.

No. Features	**Accuracy**	**AUC score**
2	70.38 %	0.7039
4	56.34 %	0.5643
4	58.22 %	0.5827

Table 3.4: Worst typical testors SVM training accuracy (Star Classification)

In contrast to table 3.3, table 3.4 shows that the selected worst typical testors present a largely inaccurate classification of the objects in the testing set. Both the accuracy and AUC scores, show that these typical testors are not able to distinguish elements from different classes. Notice that both tables, 3.3 and 3.4, display typical testors with the same length. Thus, the number of features does not seem to be a relevant factor when it comes to obtaining acceptable classification scores.

3.2.2 Star Classification Dataset (Neural Network)

No. Features	Accuracy	AUC score
11	99.54 %	0.9995

Table 3.5: Full dataset NN training accuracy (Star Classification)

No. Features	Accuracy	AUC score
4	99.61 %	0.9999
4	99.66%	0.9998
4	99.64%	0.9993

Table 3.6: Best typical testors NN training accuracy (Star Classification)

In a similar way, tables 3.5, 3.6, present great results in terms of the accuracy and AUC score in comparison to the model trained with the full set of features. Therefore, we can argue that there are typical testors that successfully reduce the number of necessary features to distinguish elements from different classes.

No. Features	Accuracy	AUC score
2	59.05 %	0.6370
4	54.94%	0.5488
4	58.71%	0.6059

Table 3.7: Worst typical testors NN training accuracy (Star Classification)

Table 3.7 corroborates the results from the SVM model since the typical testors

classified as “worst”, are not able to provide an accurate classification of the objects in the testing set.

Having analyzed the results from the first dataset, we look to further evaluate the selection algorithm with other datasets. This way, we can convince ourselves that the preliminary results obtained from the Star Classification Dataset were not biased in any way and that the method is useful for any other dataset.

3.2.3 QSAR Biodegradation Dataset (SVM Model)

No. Features	**Accuracy**	**AUC score**
41	91.77 %	0.9169

Table 3.8: Full dataset SVM training accuracy (QSAR Biodeg)

No. Features	**Accuracy**	**AUC score**
17	92.21 %	0.9199
17	92.21%	0.9202
14	87.23%	0.8710

Table 3.9: Best typical testors SVM training accuracy (QSAR Biodeg)

This dataset presents great accuracy and AUC scores throughout the best typical testor selection list in comparison to the scores obtained for the full dataset. The first two typical testors in table 3.15 have even better accuracy and AUC scores than the ones showed in table 3.8. However, the third typical testor’s scores decreased, although not a large amount in comparison to the full set training.

No. Features	Accuracy	AUC score
15	89.39 %	0.8935
14	89.17%	0.8911
15	89.61%	0.8951

Table 3.10: Worst typical testors SVM training accuracy (QSAR Biodeg)

From table 3.10 we can see that the scores obtained are not that far from the ones we got from the best typical testors. Notice that the third best typical testor presents even slightly worse scores, however the scores are also not far from the ones obtained in table 3.8. This fact tells us that, in a general manner, the typical testors obtained from this dataset will achieve good results in terms of accuracy and AUC scores when using and SVM model.

3.2.4 QSAR Biodegradation Dataset (Neural Network)

No. Features	Accuracy	AUC score
41	93.20 %	0.9437

Table 3.11: Full dataset NN training accuracy (QSAR Biodeg)

No. Features	Accuracy	AUC score
17	90.23 %	0.9379
17	89.94%	0.9384
14	88.53%	0.9268

Table 3.12: Best typical testors NN training accuracy (QSAR Biodeg)

Once again, the results obtained from tables 3.11,3.12, show that the selected typical testors successfully reduced the number of features needed, specially when we consider the AUC score. The AUC score obtained for the three best typical testors is just slightly less than the one obtained for the full set of features. It is important for us to notice that for the Neural Network, the third testor presents results very similar to the ones obtained for the first two, as opposed to what we saw with the SVM model. Thus, we can argue that the results of the selection algorithm may vary depending on the classification model used.

No. Features	**Accuracy**	**AUC score**
15	86.83 %	0.9003
14	86.69%	0.8893
15	87.11%	0.9044

Table 3.13: Worst typical testors NN training accuracy (QSAR Biodeg)

In table 3.13 we can observe similar results to the ones obtained for the SVM model. Even though these results are slightly worse in comparison to the ones in 3.12, they still maintain good AUC scores, having two with a value over 0.9. Therefore, we can deduce, once again, that typical testors maintain a good classification attribute in a general manner. Having said this, we can notice that, even this case, the best testors (only the first two in both cases) present slightly better results, closer to the one obtained from the full set of features, in terms of accuracy and AUC score. However, the selection of the worst typical testors lacks accuracy. Nonetheless, the application of this method would focus on finding the best typical testors.

3.2.5 Credit Approval Dataset (SVM Model)

No. Features	Accuracy	AUC score
15	86.17 %	0.8627

Table 3.14: Full dataset SVM training accuracy (Credit Approval)

No. Features	Accuracy	AUC score
8	84.19 %	0.8426
9	84.58%	0.8471
9	69.56%	0.6935

Table 3.15: Best typical testors SVM training accuracy (Credit Approval)

For the final dataset, tables 3.14,3.15 show similar accuracy and AUC scores, but only with the first two best typical testors. The third typical testor in table 3.15 shows a considerable deficit in both accuracy and AUC score. However, let us notice that the number of typical testors obtained in this dataset is only six. Therefore, as long as there isn't any typical testors within the "worst" typical testors list that presents results similar to that of the ones obtained with the full set of features, we can say that the algorithm is working appropriately.

No. Features	Accuracy	AUC score
9	76.67%	0.7645
9	69.96%	0.6981
9	76.28%	0.7606

Table 3.16: Worst typical testors SVM training accuracy (Credit Approval)

As mentioned before, we were expecting to not find any good results in table 3.16. We can now say that the algorithm has successfully ordered the list of typical testors, placing the best at the top. Both the accuracy and AUC scores in table 3.16, show the inaccuracy of the classification model trained with these typical testors.

3.2.6 Credit Approval Dataset (Neural Network)

No. Features	**Accuracy**	**AUC score**
15	86.36 %	0.8636

Table 3.17: Full dataset NN training accuracy (Credit Approval)

No. Features	**Accuracy**	**AUC score**
8	83.33 %	0.8408
9	83.67%	0.8184
9	79.87%	0.7360

Table 3.18: Best typical testors NN training accuracy (Credit Approval)

In a similar way, the results obtained with a Neural Network shown in tables 3.17,3.18, lets us observe that only the first two typical testors are able to train the model with a similar outcome in comparison to the full set of features. However, the third typical testor presents increased scores than the ones obtained with the SVM model.

No. Features	**Accuracy**	**AUC score**
9	79.65 %	0.7867
9	78.14%	0.7705
9	80.74%	0.7621

Table 3.19: Worst typical testors NN training accuracy (Credit Approval)

Table 3.19 shows results similar to the ones obtained for the third best typical testor. The scores obtained from these typical testors show a considerable decrease in comparison to the scores of the full set of features, specially in AUC score. Thus, the algorithm, once again, has been able to select the best typical testors for both classification models.

Having presented these results, we can now give a deeper evaluation of the effectiveness of the proposed selection algorithm. Just as well, we can further analyze the limitations of the method and its applications. In the following section, we will expand on these topics and talk about further improvements that can be made to the algorithm.

Chapter 4

Conclusions

From the results shown in the previous chapter we can evaluate the proposed algorithm for typical testor selection and discuss some of the limitations it may present. As seen throughout chapter 3, the proposed "best" typical testors displayed accuracy and AUC scores very similar to the ones obtained from the full set of features, for both the SVM model and Neural Network. Therefore, the algorithm was successfully selecting typical testors that were able to train a classification model with great results in comparison to the ones obtained by using the full set of features. This fact was seen for all three selected typical testor in the first dataset, but only with the first two typical testors for the second and third dataset. In the case of the latter dataset, the total amount typical testors was the main cause for the low scores presented by the third typical testor.

One thing that we should notice from the tables displaying the results for the worst typical testors, is that the scores obtained were somewhat ambiguous.

One may see that the values for accuracy and AUC scores didn't really hold a relationship between these "worst" typical testors. For example, in table 3.16, the first and third typical testors presented increased values of accuracy and AUC score in contrast to the second one. However, these typical testors are categorized as not useful, which is why the fact that the worst typical testors may still present certain complications in terms of the order, is not relevant. From the results obtained, we may consider the algorithm a success when it comes to focusing on finding typical testors that maintain a similar outcome for the classification model training as the one done by using the full set of features.

Furthermore, note that the length of the typical testors selected as "best" has no clear correlation their selection. In other words, the longest typical testors weren't always selected as "best". Take, for instance, the typical testors selected for the Credit Approval Dataset. The best typical testor was a length of 8 features, while the ones selected as "worst" had a length of 9 features. Likewise, in the QSAR Biodeg Dataset, the largest typical testor had a total of 19 features, while the best ones selected had only 17. Thus, even though one may argue that larger typical testors contain more information and will have better results when training a classification model, our algorithm has been able to find that the length of a typical testor doesn't necessarily affect their classification capabilities.

In essence, the proposed algorithm has been able to successfully pinpoint some of the best typical testors in each dataset, with certain flaws, specially for the worst typical testors. However, as a first approach to typical testor selection, the results obtained have allowed us to find typical testors with both accuracy and AUC scores

very similar to the ones obtained from the full set of features. In addition to this, we have been able to determine the fact that the length of a typical testor does not directly influence its capability to properly train a classification model. Moreover, some new testing must be done in order to keep improving the algorithm.

4.1 Future Work

As mentioned before, we have obtained preliminary results on this selection algorithm and it still has a lot of room for improvement. In the first place, for this work, we have only considered datasets with two classes, which is more convenient for the typical testor calculations. However, it would be essential to test the algorithm with datasets that contain various classes. In a similar way, as seen in chapter 2, we have given the same weight to the normalized similarity and dissimilarity scores in order to rank the typical testors. A variation of these weights, in other words, giving more relevance to one of these scores, may prove to be more effective when selecting the typical testors. Furthermore, for datasets that present larger sets of typical testors, it would be relevant to test the classification models with a longer list typical testors in order to contrast the information. Lastly, for cases such as the QSAR Biodeg Dataset, in which both the best and worst typical testors showed good accuracy and AUC scores, it would be pertinent to contrast these scores with those obtained for randomly selected typical testors. This would allow us to observe that, as expected, every typical testor has similar scores.

As we have seen in this chapter, although the preliminary results showed great

promise, there still is room for improvement. We will continue to develop this idea with different datasets and different approaches to testing. However, the road ahead is promising for this development.

Bibliography

[1] A. Gallego, D. Torres, F. Álvarez, and A. Torres. Identificación de características de células de cáncer de mama por medio de testores típicos. *Research in Computing Science*, pages 43–54, 2017.

[2] M. Lazo-Cortes, J. Ruiz-Shulcloper, and E. Alba-Cabrera. An overview of the evolution of the concept of testor. *Pattern Recognition*, pages 753–762, 2001.

[3] E. Alba-Cabrera, S.R. Godoy-Calderón, and J. Ibarra-Fiallo. Generating synthetic test matrices as a benchmark for the computational behavior of typical testor-finding algorithms. *Pattern Recognition Letters*, pages 46–51, 2016.

[4] J. F. Martínez, J. A. Santos, and A. Carrasco. Feature selection using typical testors applied to estimation of stellar parameters. *Computación y Sistemas*, pages 15–23, 2004.

[5] R. A. Vásquez and S. Godoy-Calderón. Using testor theory to reduce the dimension of neural network models. *Research in Computing Science*, pages 93–103, 2007.

[6] E. Alba, J. Ibarra, S. Godoy, and F. Cervantes. Yyc: A fast performance incremental algorithm for finding typical testors. *Iberoamerican Congress on Pattern Recognition*, page 416–423, 2014.

[7] A. Mammone, M. Turchi, and N. Cristianini. Support vector machines. *WIREs Computational Statistics*, pages 283–289, 2009.

[8] D. K. Srivastava and L. Bhambhu. Data classification using support vector machine. *Journal of Theoretical and Applied Information Technology*, pages 1–7, 2010.

[9] E. Weinan, C. Ma, S. Wojtowytsch, and L. Wu. Towards a mathematical understanding of neural network-based machine learning: What we know and what we won't. *CSIAM Transactions on Applied Mathematics*, page 561–615, 2020.

[10] J.C. Garcia. Mathematical neural networks. *Axioms*, page 1–18, 2022.

[11] S. Raschka. *Python Machine Learning: Unlock Deeper Insights into Machine Leaning with This Vital Guide to Cutting-Edge Predictive Analytics*. Packt Publishing, 1 edition, 2015.

[12] M. Hossins and M.N. Sulaiman. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, pages 1–11, 2015.

[13] J.N. Mandrekar. Receiver operating characteristic curve in diagnostic test assessment. *Biostatistics for Clinicians*, pages 1315–1316, 2010.

[14] Fedesoriano. Stellar classification dataset - sdss17, 2022. Last accessed 21 September 2022, https://www.kaggle.com/datasets/fedesoriano/stellar-classification-dataset-sdss17.

[15] K. Mansouri, T. Ringsted, D. Ballabio, R. Todeschini, and V. Consonni. Qsar biodegradation data set, 2013. Las accessed 13 April 2023.

[16] J.R. Quinlan. Credit approval dataset, 1987. Last accessed 19 April 2023.