POLITECNICO DI TORINO

Master's Degree in Biomedical Engineering

Machine learning-driven prediction of chemical compound sweetness based on molecular descriptors

Thesis Supervisor

Prof. Marco Agostino Deriu

Candidates

Juan Antonio di Lorenzo

Co-supervisors

Lorenzo Pallante

Marta Malavolta

ACADEMIC YEAR 2021/2022

Abstract

Taste is a complex sensation related to the perception of food flavours, acting as a key control mechanism to defend the organism from poisons or noxious substances: the five basic taste sensations, namely bitter, sour, salty, sweet and umami, evoked either alone or in combination by food molecules, may stimulate the intake or rejection of the latter. Sugars (e.g., glucose, sucrose) constitute a prominent example of this mechanism, inducing a powerfully motivating sensory response toward food in light of their role as one of the primary energy sources for the body. On the other hand, the well-known correlation between sugar abuse and diseases development, e.g., diabetes, obesity or cardiovascular problems, is driving the tendency to substitute natural sugars with low-calorie sweeteners, to promote an overall healthier lifestyle. This consideration is spiking the interest in the development of artificial sweet molecules with a low calorie content but a strong sweetening ability, a process that can greatly benefit from the molecular-level identification of key chemical and physical features that ultimately result in a specific gustatory sensation.

In this context, machine learning (ML) models can represent fast and readily deployable tools for the discovery of novel sweet compounds. With this goal in mind, in the present work, a novel set of both open-source and proprietary relevant molecular descriptors were extracted, starting from a database of 316 sweet molecules. After a thorough assessment of the state of the art, two ML models were constructed for the evaluation of the level of sweetness of a given query molecule. Given the importance of assuring the reliability of the predicted result, which can be jeopardized by the limited number of available molecules with known sweetness levels used to construct the models, an *applicability domain* of the sweetness predictor was also developed and is reported, taking the specific usage scenario into account. Overall, the present work provides a solid starting point for the future refinement of regressors as molecular predictors of the sweetness level of novel chemical compounds.

1 Introduction

As previously mentioned, the increasingly important role of the search for new molecules in the food industry has led to the increased use of molecular predictors. A sweet predictor can be used not only in the food field to discover new molecules, but for example, it can be shown to be helpful, in conjunction with other methods, to map the organoleptic characteristics of an ingredient. Currently, this situation is facilitated by the increasing computational power available and the greater tendency to implement artificial intelligence (AI) models to solve complex problems. The following thesis work is based on using a predictor that can determine the sweetness of any organic molecule and evaluate the applicability and reliability of the result obtained in the context of use.

The present thesis work is organized in 4 main Chapters

Chapter 1 is the present introductory section

Chapter 2 is dedicated to explaining the prior biological knowledge behind this work. It will mention how the taste sensor is made up, briefly review the sweetness ligands, and how the level of sweetness is measured in the lab.

Chapter 3 focuses on methods regarding data analysis, specifically the construction of the sweet database, feature selection, the machine learning models used and their optimization. Last but not least, the study of the applicability domain defines whether the predictor obtained value it's possible to apply to a given circumstance and how reliable the predicted value is.

Chapter 4 focuses on the development of the machine learning driven algorithms to predict sweet taste. In this section, employed methods and obtained results will be discussed. A conclusion will point out main findings, limitations of the work and further future developments.

2 Biological and chemical background

2.1 Sweet taste receptor

The five primary basic tastes are umami, bitter, salty, sour, and sweet. The sensation of sweet, umami and bitter tastes are determined by organic molecules called G protein-coupled receptors (GPCRs), while sour and salty are determined by ion channels. In the following, we will briefly mention the receptor that provides the sensation of sweetness and then focus on the molecules that cause the phenomenon.

The receptor that recognizes sweet substances such as sugars is a heterodimer composed of two subunits TAS1R2 and TAS1R3 which are encoded in our genes: tas1R2 and tas1R3. Looking at Figure 1 from the bottom, we can see the following composition: a secondary structure with 7 transmembrane helices (TMD), an extracellular N-terminus composed of a Venus flytrap module (VFTM), and a cysteine-rich domain (CRD) attached to the transmembrane domain.

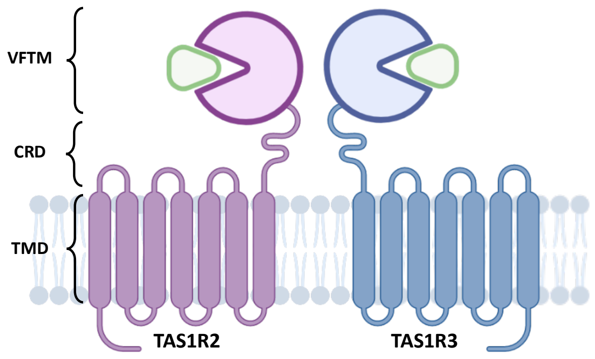


Figure 1 – Sweet receptor [1]

This receptor responds to many natural and artificial compounds (e.g., Cyclamate, Monellin, Glucose, Sucrose). Therefore, the sweet receptor contains different active sites within a single VFTM (usually binding small sugars) and sites formed using the VFTMs of both TAS1R2 and TAS1R3 subunits (glucose is an example of a molecule using such a site). Finally, we can find allosteric binding sites within the transmembrane core of the TAS1R3 subunit that activate the sweet receptor (for example, the site is activated by cyclamate). It is also important to note that some binding molecules can play an antagonistic role to the sweet molecules (lactisol is one such molecule).

These examples are intended to show how there are many specific activatable sites that are activated by molecules with as many specific characteristics. This work will aim to extrapolate the chemical characteristics of the stimulating molecules through molecular descriptors and construct a predictor capable of determining whether a molecule under consideration possesses the properties necessary to activate the receptor mentioned above.

2.2 Sweet ligands and their sweetness level

Since the first classifications of molecules in 1976, the sweet molecules database has expanded very quickly to include several proteins recently [2]. This situation makes us realize that there are many compounds that can have a sweet taste from different categories.

The Figure 2 shows some sweet molecules and their structure: from elementary molecules on the top to more complex compounds on the bottom.

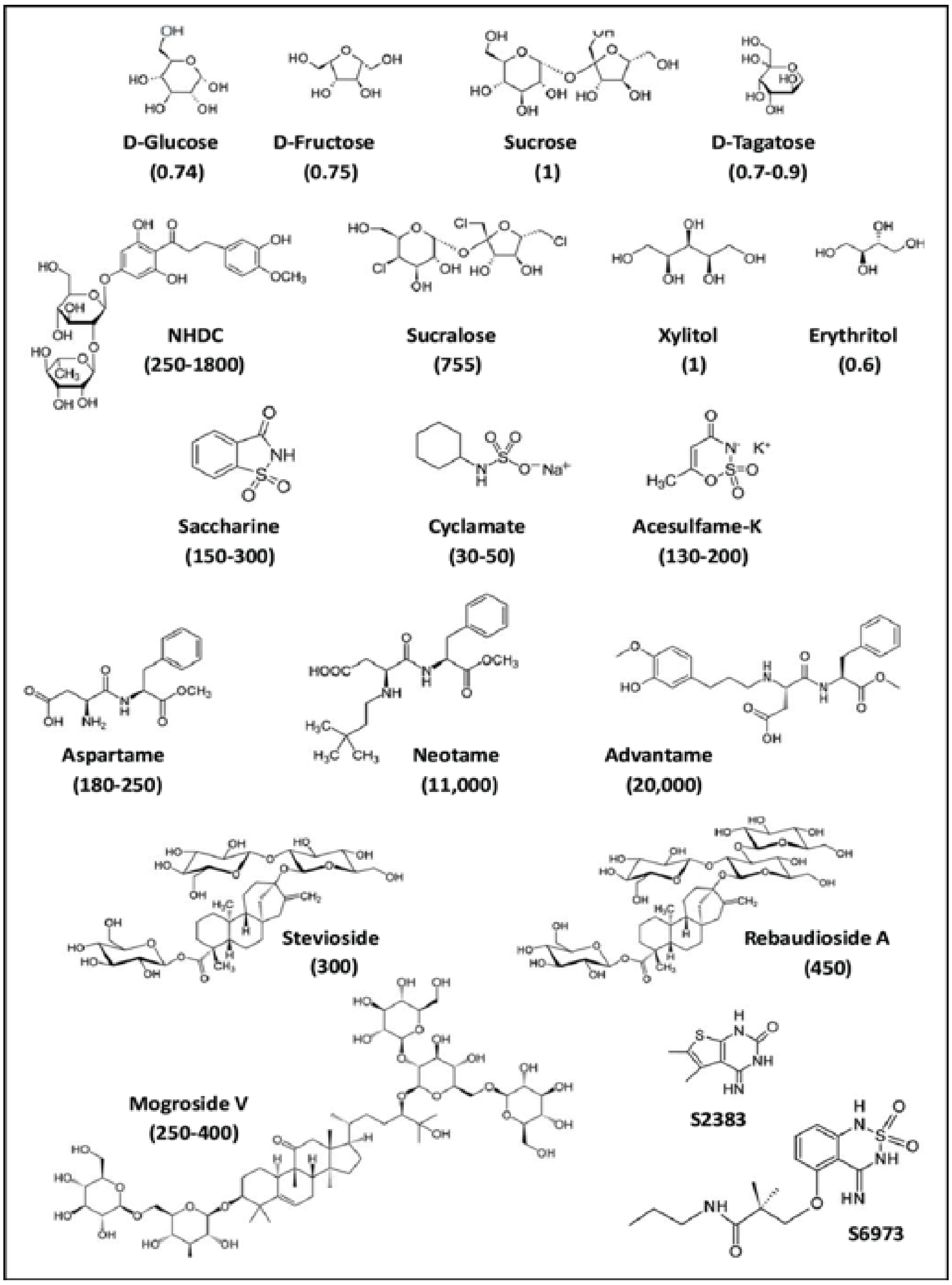


Figure 2 - examples of structures of sweet molecules [3]

Before proceeding further, it would be helpful to briefly understand how the level of sweetness is assessed in the laboratory. At present, the results obtained are considered to be actual and are the benchmark for all other measurement methods.

We will briefly explain the process used in the laboratory. The first step is to obtain cells expressing the sweet human receptor. This detailed procedure is described by Poirier et al. [4] and we will mention it briefly. The cDNA encoding TAS1R2 and TAS1R3 subunits are taken and inserted into two vectors, pcDNA3 and pcDNA5, respectively. Using HEK293T cells stably expressing the chimeric G protein subunit Gα16gust44, which is first treated and subsequently cotransfected with previously mentioned plasmids encoding the human sweet taste receptor subunits hTAS1R2 (in pcDNA5/FRT, Invitrogen) and hTAS1R3 (in pcDNA3, Invitrogen). After obtaining a cell culture expressing a human receptor, the calcium level is measured, which correlates with the intensity of the stimulus received by the cell. A luminescence assay is used to assess the stimulus level.

In Figure 3, a possible spectrum of stimulation and emission is shown.

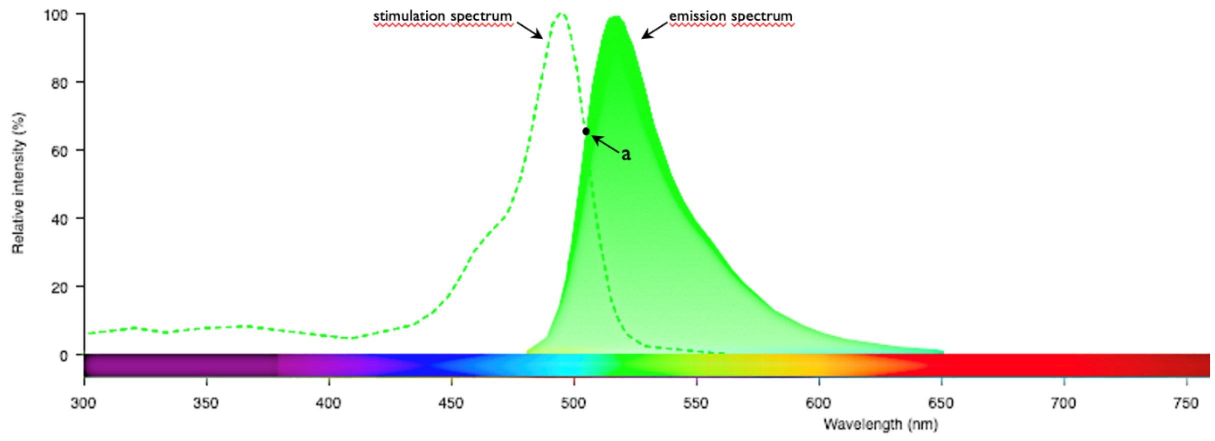


Figure 3 – Spectrum Flu4-am [5]

The fluorophores we mentioned are molecules that show an increase in fluorescence when they bind to Ca2 up to 100 times. This behaviour leads to a concentration/response curve similar to a sigmoid. To measure the trend, two extreme values must first be measured. The first is value in the absence of stimuli, and the second is value in saturation. Figure 4 below shows the behaviour obtained in the laboratory by Bouysset and his collaborators for sucrose (a) and arctiin (b) [6].

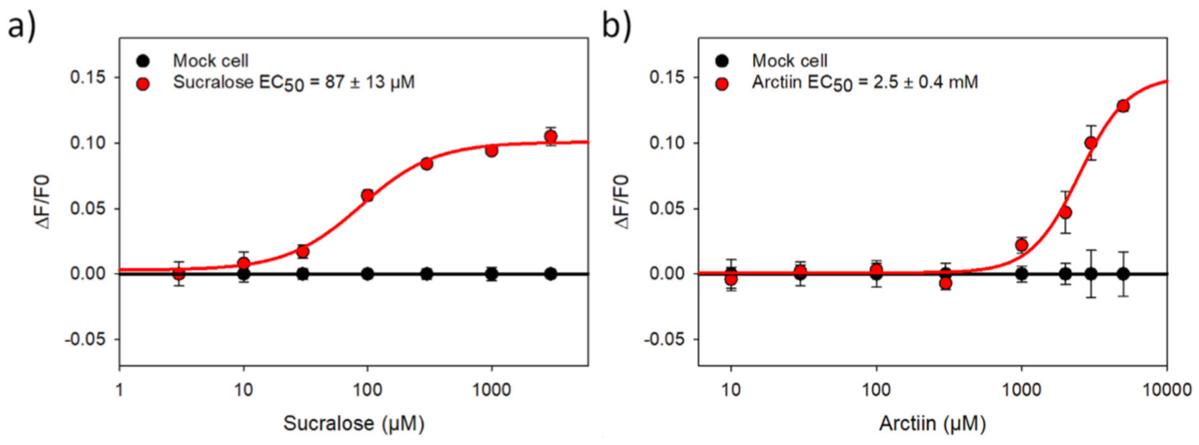


Figure 4 – Example of sweet level measurement in the laboratory

The black line corresponds to the control sample, where the cells were transfected with an empty plasmid (no gene encoding TAS1R2 and TAS1R3 subunits were inserted).

Finally, to assess the level of sweetness, we take the concentration value that produces a stimulus equal to 50% of the maximum stimulation (EC50). The equation below shows the calculation performed:

$$f(x) = min + \frac{max - min}{1 + \left(\frac{x}{EC_{50}}\right)^{-Hillslope}}$$

3 Materials and Methods

3.1 Database construction

3.1.1 Sweet molecules

A sweet molecule is defined as a molecule capable of stimulating sweetness. At first, it was thought that molecules belonging to this category must have very similar structures, but as shown in Figure 5 this is not true.

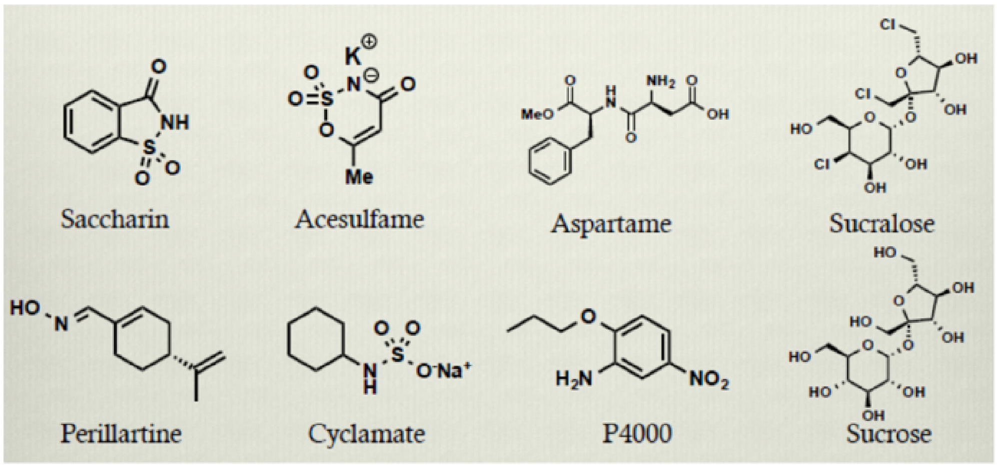


Figure 5 - Examples of different sweet structures

However, the molecular conformation is not the only factor to be considered to explain the biochemical activity. Another aspect to consider is the bonds that molecules and receptors create. In 1963, Shallenberger formulated the "AH-B model" theory. It was one of the first developed, attempting to explain why molecules with different structures were all sweeteners [7].

Briefly, the model highlights that receptors are characterized by two functional groups: a hydrogen donor (-AH) while the second is a hydrogen acceptor (-B). Similarly, there must be two functional groups in the sweet molecule having the same characteristics as those present in the receptor. This combination leads to the creation of a bond, as shown in Figure 6.

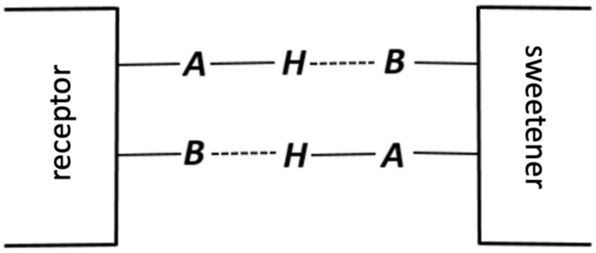


Figure 6 - Bonds in the AH-B model

Currently, with the wide use of computational technologies, there has been a shift from creating a single prediction model to extrapolating a set of molecular features that would allow them to be correlated to the output.

Therefore, a fundamental step for the creation of a prediction model is the collection of a database containing molecules of which the level of sweetness is calculated in the laboratory, using the method of a cellular luminescence assay briefly mentioned in *chapter 2.2*. Currently, this process represents the state of the art, so these values are considered the actual sweetness values and taken as a reference point.

Given the complexity and the cost of a luminescence analysis process, not to mention the previous knowledge needed to carry out such a process, very few databases (DB) are available, and one of these is the one generated by Cheron's work called *SweetenersDB* [8]. The DB has been updated in the following years, introducing new compounds (Ruiz-Aceituno et al., 2018). to a total of 316 molecules.

The DB has, for each element, the name of the compound, its graphical representation and finally, the level of sweetness expressed in logarithmic form. Figure 7 shows the characteristics of 2 molecules:

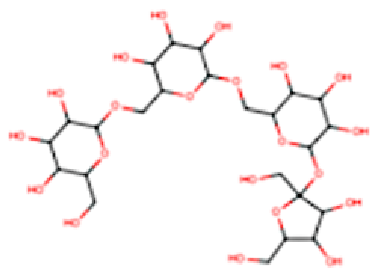
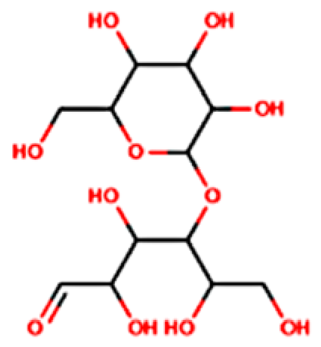
Name	representation	Log(S) ▲
stachyose		-0.66
lactose		-0.6

Figure 7 - Two molecules belonging to SweetenersDB are shown

Molecules are classified by name, while the calculation of the sweetness value is relative to the sweetness value of sucrose. So the latter, in our case, is given a sweetness level of 0. In other molecules, the relative sweetness value is defined as the ratio of the concentration of a sucrose solution to the concentration of a solution of the molecule under consideration.

In the literature, other databases are created by the collection of molecules, such as the work of Ahmed et al. [10] called SuperSweet, which has about 8000 items and is the most extensive current collection of sweet molecules. However, this database is not downloadable, and Chéron et al. [8] conclude that 99% of the molecules present in SuperSweet are close to those in the SweetenersDB database. Moreover, the DB created by Chéron (Chéron et al., 2017). is widely employed in other works, thus facilitating an immediate comparison.

For all these reasons, we have chosen a database that provides us with a solid starting point despite the not too high number of molecules.

3.1.2 Molecular descriptors

Another component underlying our research is to find a relationship between the structures of a molecule and its chemical properties. About the latter, we generally mean all properties that can affect a molecule: these can be physical, chemical or even biological such as the evaluation of the pharmacological activity.

The mathematical objects can usefully and unambiguously describe the chemical structure of a compound; these are the descriptors. The use of the latter is important because they allow one to predict a priori whether a molecule can have a specific characteristic without actually having to synthesize the structure of the molecule.

A first distinction of the molecular descriptors, being mathematical objects, is made based on the spatial order of the structure from which they are derived so that we can have different descriptors such as 1D, 2D and 3D. The 1D descriptors are obtained starting from the brute formula while, as it can be guessed from the word name itself, the other two (2D and 3D) from the representation of the structure in two- and three-dimensional format, respectively.

In this section, we will discuss descriptors extracted from open-source modules and MOE software.

The first open-source module we are going to analyze is Chemopy [11], freely accessible and written in Python. It consists of 19 categories and includes 1135 descriptors; in the table present Appendix 1we can see the categories to which the descriptors belong. For the complete list, see the link in the work of Cao [11]. Remember that this module requires the installation of the following external packages: RDKit, Openbabel, MOPAC and Pybel.

The second open-source module is Rdkit [12]. It consists of 208 descriptors belonging to different categories listed in the table in Appendix 2. This module is written in Python and C++, whose strength is the speed of execution of programs. Please note that Rdkit is not only a module for descriptors, but it is an entire ecosystem composed of several packages to manipulate molecules. In fact, the creation of molecular structures starting from Simplified Molecular Input Line Entry System (SMILES) in molecular structures is done using one of these packages.

Mordred [13] is the latest open-source module used for extraction; it can extract up to 1826 descriptors. The entire module is written in Python and is available with the Conda environment [14] environment and is the most extensive collection of descriptors. Appendix 3 is also provided for the latter, listing the category to which all descriptors belong.

Turning to the MOE software provides the possibility of extracting up to 440 descriptors. Note that the MOE software before extracting the descriptors allows to perform an energy minimization of the structure. As we will see later, it can be an advantage in terms of performance. The list of the family of descriptors extracted by MOE is available in Appendix 4.

3.1.3 The use of molecular descriptors to extract sweet molecules properties

"Data pre-processing involves transforming raw data into well-formed datasets so that a type of analysis can be applied. Raw data is often incomplete and has inconsistent formatting. The adequacy or inadequacy of data preparation has a direct correlation to the success of any project involving data analysis."

Starting from the definition of pre-processing, we can immediately understand how fundamental the antecedent processing of the data is. We start from the database with the molecules saved in SMILES format. [15]. In fact, to have a compact database, it is chosen to use the Simplified Molecular Input Line Entry System (SMILES) format because it allows a considerable saving of memory. It is constituted by a string essentially formed by characters codified in the American Standard Code for Information Interchange (ASCII) format. This ensures the saving of memory and interoperability between different systems (software, operating systems).

The conversion from SMILES to molecular structure and vice versa follows precise rules found in Anderson's work [15]. We will briefly mention them below.

The first rule deals with Atoms and Bonds and states how a molecule should be represented in string form, the letters for each atom, and the symbols representing the different types of bonds. The second rule deals with simple chains, providing guidance on how bonds between structures should be represented, considering the suppression of hydrogen atoms (the latter are not represented). The last three rules concern the representation of branches, rings and atoms along the chains.

After obtaining the confirmation, it is necessary to uniform the structure, applying Standardisers. The last ones are a set of operations that filter and transform the molecules to agree with a well-defined set of rules. We report below in Table 1 the rules used in general by Standardisers, both Flatkinson and MOE.

Table 1 - Standardization rules applied by Flatkinson and MOE

APPLIED RULES		
Number rules	Flatkinson	MOE
1	Bond breaks between molecule and group I and II metals	Bond breaks between molecule and group I and II metals
2	Removal of salt molecules not belonging to the structure	Removal of salt molecules not belonging to the structure
3	Neutralization of the structure itself	Neutralization of the structure itself
4		Energy minimization is possible

At this point, it is possible to calculate the descriptors.

3.2 Machine learning models and their optimization

The machine learning models that we are going to analyze are called ensemble methods. In particular, we will explain the amplification method called Boosting. These are different from the methods called Bagging, where at the end of the implementation of the various models, a majority vote is performed to obtain the result. Instead, in Boosting, the concept is to focus on samples challenging to classify to improve overall performance. However, these methods bring a disadvantage, the tendency to have high variance and consequently overfitting on the training data. [16].

Adaptive Boosting (AB) [17] and the Gradient Boosting Regressor (GB) [18] are two models that we will discuss in the following the steps of the Boosting method are shown in Figure 8. Then the substantial differences between the two chosen networks are analyzed.

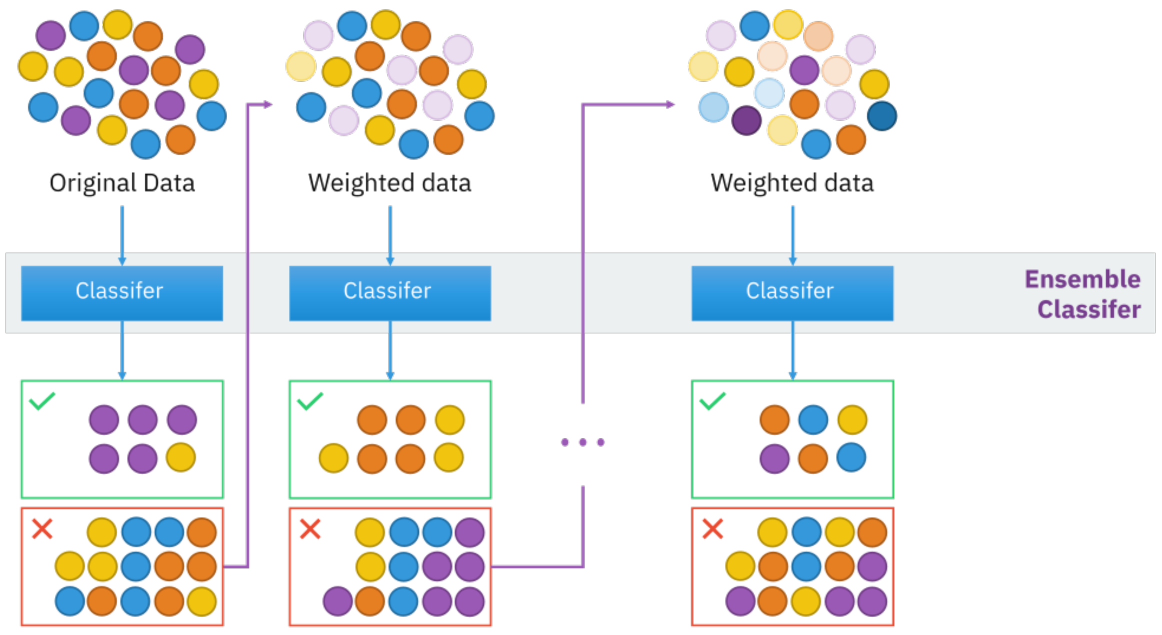


Figure 8 - Boosting process [19]

Figure 8 highlights that each step corresponds to the training of a model, which focuses on the data whose prediction was incorrect (red box).

The differences among the chosen networks concentrate above all on how the weights associated with every created model are updated. The difference resides above all in the function of Loss that is possible to choose in the GB, while in the AB, it is exponential.

As far as network optimization is concerned, first of all, one has to agree on the function (scoring) to evaluate performance. One way is to use the regression formula found in the work of Golbraikh and colleagues [20] reported below:

$$R = \frac{\Sigma(y_i - \bar{y})(\tilde{y}_i - \bar{\tilde{y}})}{\sqrt{\Sigma(y_i - \bar{y})^2 \Sigma(\tilde{y}_i - \bar{\tilde{y}})^2}}$$

where y and $\tilde{y}$ represent the observed (the actual) and predicted values, respectively.

To optimize grids, we need to search for the parameters that provide the best performance. One possible function that performs this search is the Grid Search, a Scikit-learn module [21] to which a parameter space must be provided. The Grid Search performs an exhaustive search of all combinations of parameters (estimators, laerning_rate) in the network until it obtains the one that gives it the best performance.

Note that Grid Search being an exhaustive search, the computational cost can be high, so using a search space with as few parameters as possible is preferred. It is discarding all features that have little impact on the performance of the networks.

3.3 Features selection

Feature selection allows choosing among the total features, a subset that can improve the accuracy and efficiency of the predictor. The methods for feature selection are divided into three main categories: filters, wrapper and embedded, as shown in Figure 9.

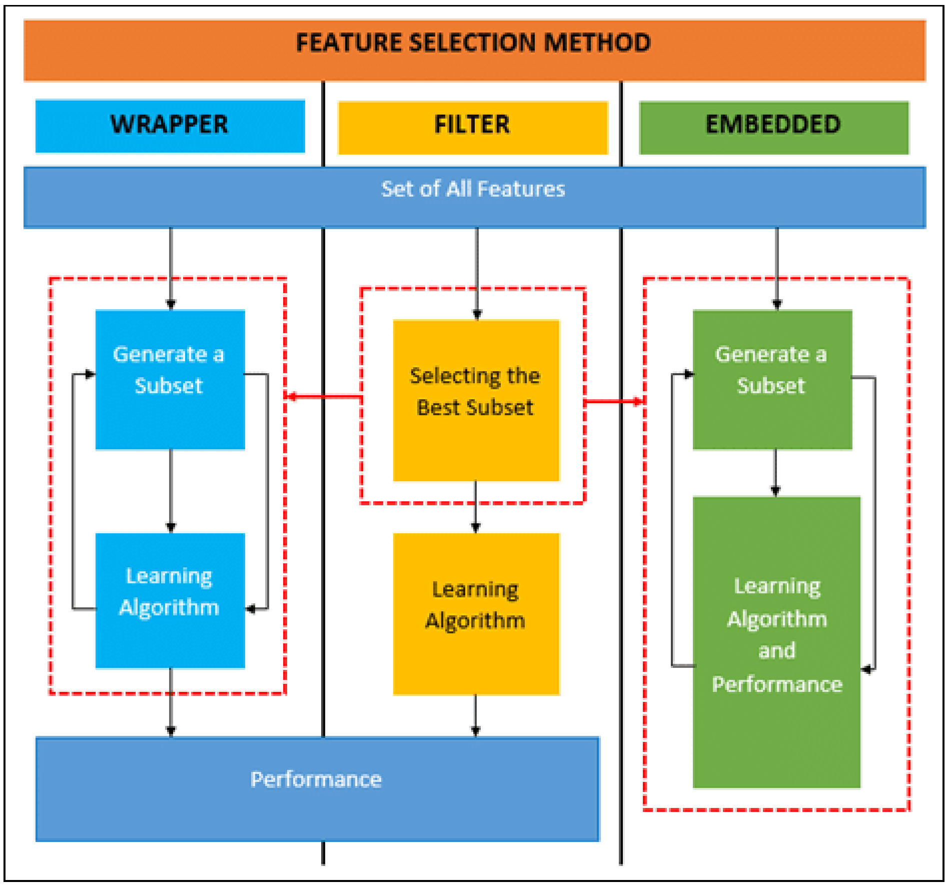


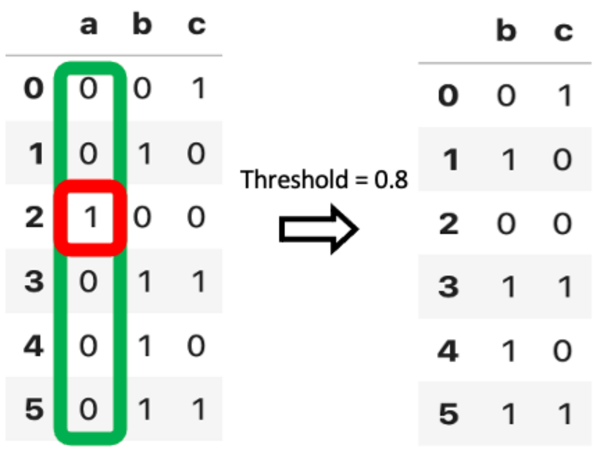
Figure 9 - Features selection methods

The method in the middle of Figure 9, filter, can be used as a pre-processing method before applying techniques belonging to the other two categories.

The Filters method carries out a pre-processing of the data, which is independent of the prediction algorithm used but is based on considerations regarding the general characteristics of the training set. Among these are the Bernoulli filter and the cross-correlation. Both are very versatile methods, not complex and very effective.

The Bernoulli filter, in particular, analyzes the variance present among the values of a feature. After setting a threshold value, this filter removes a feature if it has a higher variance. Below is an example with binary features Figure 10 where the probability is calculated using this formula:

$$Var[x] = p(1-p)$$



	a	b	c
0	0	0	1
1	0	1	0
2	1	0	0
3	0	1	1
4	0	1	0
5	0	1	1

	b	c
0	0	1
1	1	0
2	0	0
3	1	1
4	1	0
5	1	1

Figure 10 - Bernoulli filter

The example shows a feature with a probability p= of 5/6 (feature "a") higher than the threshold value and is discarded.

This filter removes all the properties with low information content. However, it would also be interesting to remove those properties that are very similar to other features despite having information content. In this case, the implementation of another filter, the cross-correlation filter, can help us. The Pearson correlation formula is used:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\ \sqrt{\sum_{i=1}^{n}(y - \bar{y})^2}}$$

where x and y are the features under consideration and the summation goes from 1 to the number of features (n).

The correlation between the features taken in pairs is calculated, and one is discarded if the correlation is higher than the threshold value.

An example of the application of both filters in a series is shown below. On the axes, the numbered features are represented. Between Figure 11(a.) and (b.), we can see the removal of the features with low variance (grey colour) and the removal of the highly correlated features (represented by red).

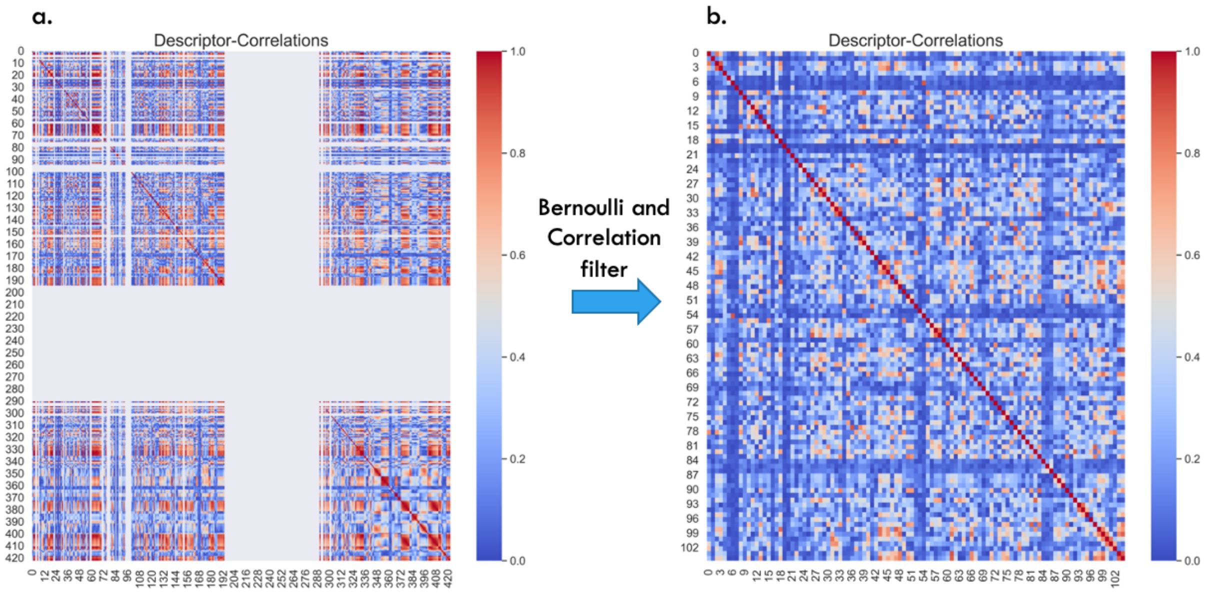


Figure 11 - Filter application, heatmap before (a.) and after (b.)

After filtering, we could either use the prediction algorithm or further select by choosing one of the other two methods (wrapper and embedded).

The wrapper method uses the construction of a prediction model to evaluate the effectiveness of a subset of features. This motive is the reason why this technique is strictly dependent on the learning algorithm used. There can be two different types of approaches, the randomized one and the deterministic one. In the first one, a set of attributes is chosen randomly (as the Genetic Algorithm does). In the second one from an original starting subset, one feature is added, removed or substituted (this procedure is used by RFE and Mlxtend instances).

Starting with the latter, Mlxtend was created by Raschka. [22], the algorithm can proceed in two different ways. The first one, called Backward, starts by analyzing the entire set of features and, at each cycle, chooses whether to exclude one. On the contrary, the forward method starts from an empty subset and, at each cycle, decides whether to include one.

They are equivalent methods; we see the steps of the forward process called Sequential Forward Floating Selection (SFFS):

1. you must give him the number of features to select (it can also be a range)
2. you create an empty set,
3. the first feature is included, and performance is evaluated.
4. The performance obtained with the addition of the feature is compared to that obtained in the previous step. If the performance with the addition of the feature worsens, the feature in question is discarded (obviously, during the first cycle, this step is not performed).
5. We proceed until the desired number of features is reached, and the algorithm has evaluated all elements belonging to the initial set.

The second deterministic wrapper method we see is Recursive feature elimination (RFE). (Pedregosa et al., 2011). We start from the complete database of features; a prediction is made. The feature importance is calculated in the next step, and the least important recruited feature is eliminated. It is possible to choose the minimum subset value to be obtained.

The third method taken into analysis is the genetic algorithm [23] which, as previously mentioned, is a random method. Genetic algorithms are heuristic methods of search and optimization; they are inspired by Charles Darwin's principle of natural selection. The algorithm tries to solve problems with changing conditions by following a finite series of standard steps. A flow chart with the main steps is depicted in Figure 12.

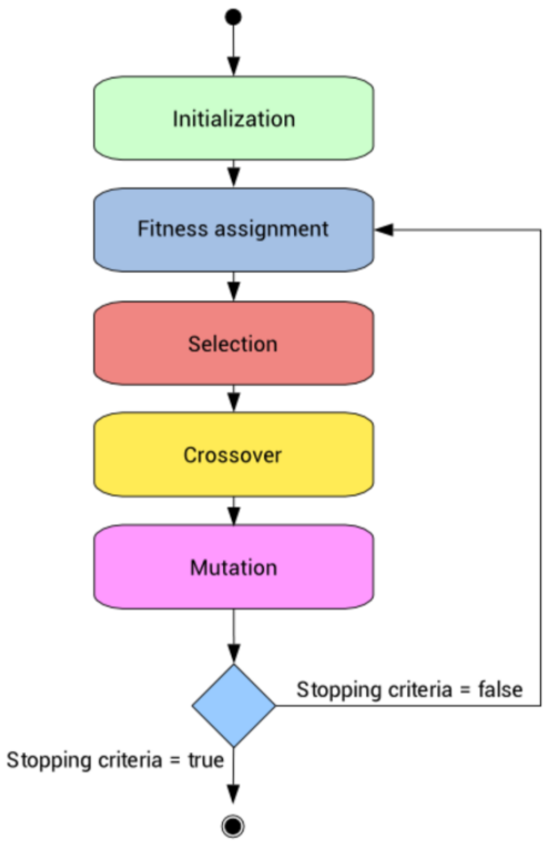


Figure 12 – Flow chart Genetic Algorithm

The steps outlined above will be briefly explained. For a complete discussion, please see Holland's paper ("Genetic Programming and Emergent Intelligence," 2020). The first step is to form the initial population. The features are divided into subgroups containing a part of the extracted features, and a feature can be included in more than one subgroup. The set of subgroups forms the initial population.

Then in the second step, the subgroups are evaluated according to a function called the fitness function. The function evaluates whether one subgroup is better than another. The third step is the selection of subgroups that lead the predictor to have the best performance. The last two steps can randomly change the result of selection through mutation and cross-over. These two steps are the fundamental part of the genetic algorithm. The mutation allows to maintain or include in a selected subgroup one or more features, while the cross-over allows mixing the features belonging to different subgroups (the number of features per subgroup does not vary).

We return to speak about the main filtering methods; the Embedded algorithm provides a middle way between the filter and wrapper methods explained. The peculiarity of this method lies in the selection of features that are included in the optimization process of the predictive model.

3.4 Applicability domain

Different methods are proposed to analyze the goodness of a prediction, but all aim to define a molecular domain. For example, in the work of Wiener [25], the domain is calculated with PCA, while in Tuwani's work, the Euclidean mean distance between molecules is calculated by selecting some features (Tuwani et al., 2019). All this makes it very difficult to compare the different works; we will analyze, in particular, the work of Hanser [27] where rather than providing methods to be used to compute a domain, he gives guidelines.

The work just mentioned provides an overview of assessing the applicability and reliability of an obtained result. Applicability is intended to consider one main issue;

is the class of my compound supported by the model?

To answer this question, we attempt to define a domain that is constructed as rigorously as possible. One of the methods this can be done is creating a Convex Hull domain, particularly using the Quick Hull (QH) algorithm [28].

The basis of this algorithm is to solve a geometric problem. In other words, given a number of points, try to find a convex domain that contains them all. As Figure 13 (a.) shows, the QH starts by dividing the number of points into two parts, one upper and one lower, and in the next step, Figure 13 (b.), it analyzes both parts separately by constructing two semi-domains. In the third step, Figure 13 (c.), the two semi-domains are merged into a total domain.

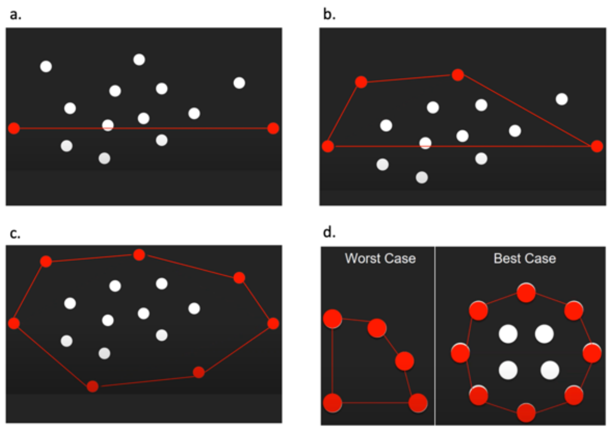


Figure 13 - Quick Hull steps

QH's strong point is its ability to find a convex domain quickly, as in the case on the right in Figure 13 (d.). However, in Figure 13 (d.), an example is shown on the left of a case where it is not possible to split the domain into two parts and this to have a more significant number of computations, from O(nlog n) to O(n^2) where n is the number of points provided. On the other hand, reliability tries to answer how reliable this result is about the context of using a predictor.

We may have situations where the confidence level must be high to ensure the goodness of the outcome, as in human safety assessment, so we set an arbitrary threshold based on experience. We take a molecule under examination, and we evaluate the number of molecules of the training set that in the domain have a distance lower than the chosen threshold. The density of information available within the domain is examined in Figure 14. The higher the density and the better the reliability of the result.

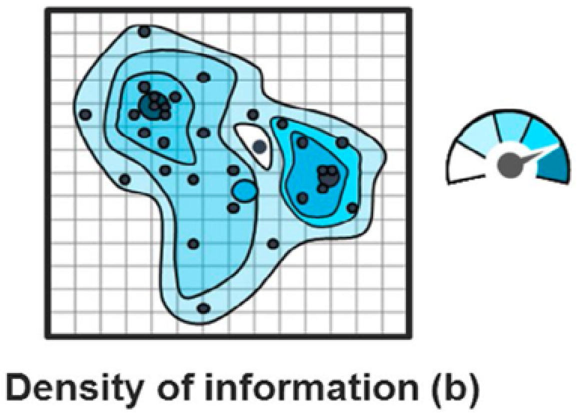


Figure 14 - Density of information

4 Machine learning-driven prediction of chemical compound sweetness

4.1 Abstract

The work of building a predictor that is capable of obtaining the sweetness value of chemical compounds has been our goal. This goal stems from the exciting applications to which the use of a predictor can lead. We started with a database of molecules that we had to standardize to be uniform. Then descriptors were calculated to extract the typical properties of a sweet molecule.

The properties were the information base on which to train our Machine Learning based predictor. After obtaining the sweetness value, we analyzed the sweetness and how valid the obtained result was based on a possible scope of use. And how accurate the obtained value could be recruited. To build our model, we took a cue from previous work [6,29,30], placing our work within the research landscape and analyzing the strengths and weaknesses in using Embedded machine learning techniques as predictors.

4.2 Introduction

The benefits of building a taste predictor are many. Several fields are touched by this research, starting with the food, public health and industrial sectors. In particular, focusing on the taste of sweet, we can say that this arouses pleasant sensations and is an instinctive means to find sources of energy (carbohydrates, usually known for their sweet taste). The matter is not only focused on finding food, but it is interesting to notice how some sweet receptors are also expressed in different organs (intestine and pancreas). Exist a strictly correlated to physiological processes, such as intestinal absorption, metabolic regulation, and glucose homeostasis (remember the primary source of energy for the brain) [31]. It is recalled that there are many food additives used to replace nutritional sweeteners such as sucrose, as consuming excessive sweet foods can lead to high risks for metabolic disorders and even cardiovascular disease. Therefore, finding molecules that can preserve sweetness without risk to human health is an ongoing challenge. To date, no artificial sweetener can accurately replicate the sweetness profile of sucrose.

So, performing an in-silico prediction of a sweetener could help quickly identify the best candidates (the most promising molecules) before conducting an expensive and time-consuming lab experiment in synthesizing the molecules. Currently, the main computational methods for sweetener prediction are based on molecular structure analysis. In recent years, several researchers have studied this field by employing different types of Machine Learning (ML) methods, such as Artificial Neural Network (ANN) [30], Adaboost Regressor [6] and Multilinear Regressor(MLR) [32]. We will analyze the steps involved in constructing the predictor and compare the results obtained with the current ML methods mentioned above.

4.3 Materials and methods

4.3.1 Data collection and data preparation

The 316 sweet molecules were imported as Simplified Molecular Input Line Entry System (SMILES), which is a method to describe the structure of a molecule using a short ASCII string. The molecules were preprocessed using proprietary software, i.e. MOE [33], or another open-source and recently-developed molecule standardizer tool, namely the Flatkinson Standardisers [34]. The former performs a disconnection of the metals of the first group into simple salts, a selection of the largest molecular fragment (elimination of salt molecules), a protonation at pH 6.9, the same as in the mouth, and the generation of a two- and three-dimensional structures. Two different data sets were obtained, one including the two-dimensional structures and one representing the three-dimensional structures of the molecules. Using the Flatkinson Standardiser, which does not have a setup like MOE but only applies a set of chemical rules, allows us to obtain a third starting dataset. The standardization allows us to have a dataset composed of molecules congruent to a group of rules. This choice gives us the possibility to reconstruct the DB always in the same way and to be able to replicate eventually the results that we will obtain.

4.3.2 Molecular descriptors

Starting from the three above-mentioned databases, the descriptors were calculated using the MOE software and the open-source Python modules Rdkit, Chemopy, and Mordred. The extracted descriptors were divided into two groups, the first containing those calculated with MOE and the second including all three open-source modules. In addition, a third group of descriptors was created, containing the 51 descriptors used in previous literature [6].

Due to the large number of molecular descriptors to be studied and the different standardisation methods that can be used, a comparison and evaluation were carried out between the 51 Bouysset descriptors (listed in Appendix 5) extracted from different Standardizers.

4.3.3 Models optimization

Before proceeding with the actual optimization of the model, we made sure to build a predictor that had a performance comparable to that currently achieved in state-of-the-art. We took as a reference point the work presented by Bouysset and colleagues [6]. We ran our two machine algorithms AB and GB, using standard parameters with the 51 descriptors from Bouysset's list. The database containing the 316 molecules was divided, as described in chapter 3, into 252 train molecules and 64 test molecules, using a 70:30 partitioning scheme as used in previous literature [6].

The parameters that manage the structure of the AB and GB grids were optimized using the Grid Search module of scikit-learn. The optimization concerned for AB the following parameters: estimators, learning rate, and Loss function. While in GB, in addition, the Depth of the network has been modified. Finally, a prediction is made again using the optimized networks.

Once the optimization of the networks is complete and we observed comparable performance with the ones achieved by Bouysset, our two networks were the starting point for the use of MOE and open-source descriptors.

4.3.4 Feature selection

Starting from the features extracted from MOE and the open-source modules, we applied two filters, i.e. Bernoulli and cross-correlation, before applying the feature selection (FS). We removed all features with a low variance with the Bernoulli filter due to the slight information content of that features. With the cross-correlation filter, it was possible to remove all the features that had a high correlation and were therefore similar. The Bernoulli filter is used first. Next, the cross-correlation filter is used, setting the threshold value to 0.92. This parameter was chosen to obtain a number of features at least equal to the number of molecules in the training set (252 features) to limit overfitting problems. In Appendix 6 and Appendix 7, it is possible to view the complete lists of selected features.

The feature selection methods applied to the databases were wrappers: we used recursive feature elimination (RFE), genetic algorithm (GA), and machine learning extensions (Mlxtend), first with AB and then with GB.

Regarding the parameters' setting, in the RFE, we set a minimum number of features equal to 10. In the GA, we chose to start with a population comprising the entire dataset. Finally, Mlxtend has been set to start from 0 and add features at each cycle, up to a maximum number of 51.

Finally, as the last step, a finer tuning of the network parameters was performed to adapt them to the selected features.

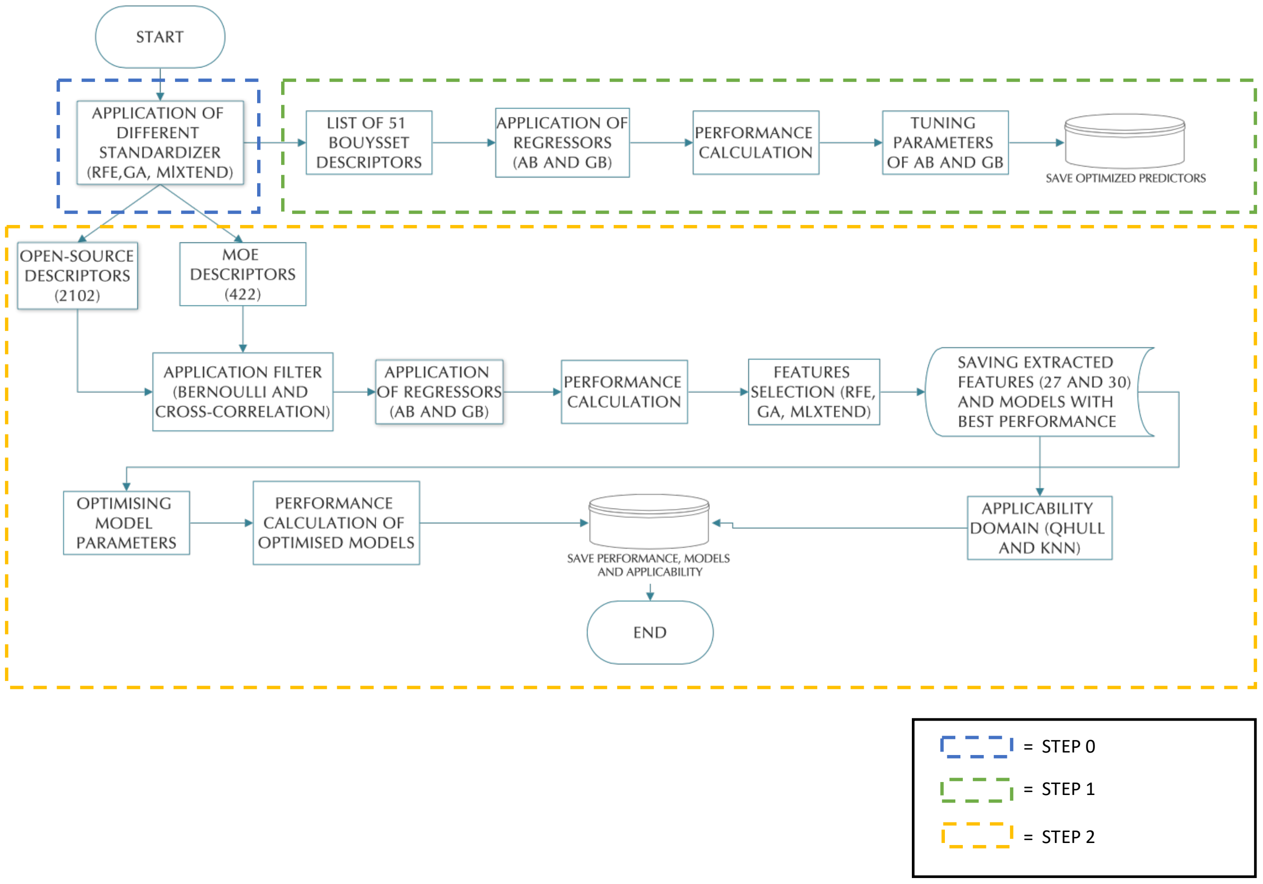
For each developed model, after extraction of the selected descriptors, an optimization of the parameters was performed in the previously mentioned way. Finally, we calculated the applicability domain.

4.3.5 Applicability and reliability domains

We used the QH algorithm to assess applicability, creating a domain formed by the molecules within the training set. The choice of not using all the available features, but only the first 10 in order of importance, is dictated by the computational cost required to create the domain. After the creation, it is possible to evaluate if a molecule under examination is inside or outside the domain, which corresponds to say if the predictor is applicable to that molecule or not.

For reliability, however, we set the KNN threshold value to 10% (arbitrarily modifiable) of the 2 molecules with maximum distance present in the domain. The distance evaluation was performed by calculating the Euclidean distance between the molecules that make up the domain. Finally, having found the pair of molecules with maximum distance, this value is taken to set the threshold value of 10%. Starting from the analysis molecule, we consider the number of molecules with a distance less than the threshold value; the higher the number of molecules, the better the reliability of the result. The agreement is normalised. If the value is 1, all molecules in the dataset are less than the threshold distance from the analysis molecule.

A flowchart of all the steps performed is shown below.


START
APPLICATION OF DIFFERENT STANDARDIZER (RFE,GA, MIXTEND)
LIST OF 51 BOUYSSET DESCRIPTORS
APPLICATION OF REGRESSORS (AB AND GB)
PERFORMANCE CALCULATION
TUNING PARAMETERS OF AB AND GB
SAVE OPTIMIZED PREDICTORS
OPEN-SOURCE DESCRIPTORS (2102)
MOE DESCRIPTORS (422)
APPLICATION FILTER (BERNOULLI AND CROSS-CORRELATION)
APPLICATION OF REGRESSORS (AB AND GB)
PERFORMANCE CALCULATION
FEATURES SELECTION (RFE, GA, MLXTEND)
SAVING EXTRACTED FEATURES (27 AND 30) AND MODELS WITH BEST PERFORMANCE
OPTIMISING MODEL PARAMETERS
PERFORMANCE CALCULATION OF OPTIMISED MODELS
SAVE PERFORMANCE, MODELS AND APPLICABILITY
APPLICABILITY DOMAIN (QHULL AND KNN)
END
= STEP 0
= STEP 1
= STEP 2

4.4 Results

4.4.1 STEP 0: Comparison of standardization methods

The effects of the two different standardization methods, i.e. MOE Wash procedure and Flatkinson standardizer, performed on the computation of the descriptors were initially evaluated. The frequencies of distribution of two different features, one representing the atomic mass (BCUTm2) and the other representing the valence of electrons (AATSC2dv), are shown in Figure 15. These two features were chosen to show that in some cases the standardization process affects the distribution of features. There are some differences in the distribution of descriptors extracted using the MOE and Flatkinson Standardisers, while there is no evidence for descriptors extracted using MOE 2D vs 3D.

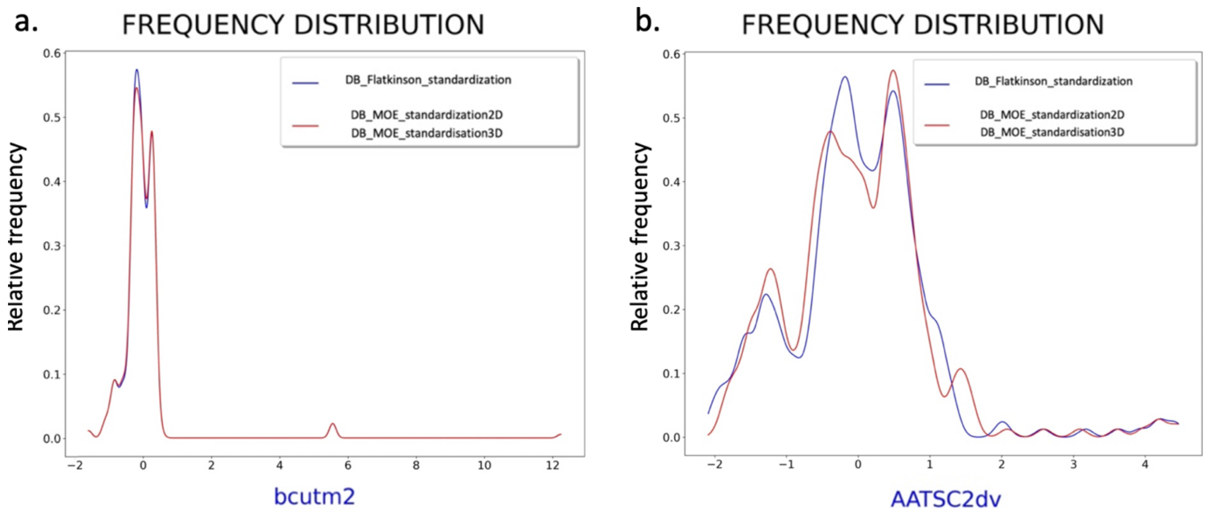


Figure 15 – Representation of Bcut2m feature distribution frequency (a.) and Representation of AATSC2dv feature distribution frequency (b.)

In order to quantify these differences, we calculated the regression values of the 51 descriptors obtained with both standardisers (MOE and Flatkinson). In Figure 16, we can observe the regressions of the two descriptors taken as examples (bcutm2 and AATSC2dv). R^2 very close to 1 indicate the similarity between the trends of the descriptors obtained with the standardization of MOE and Flatkinson.

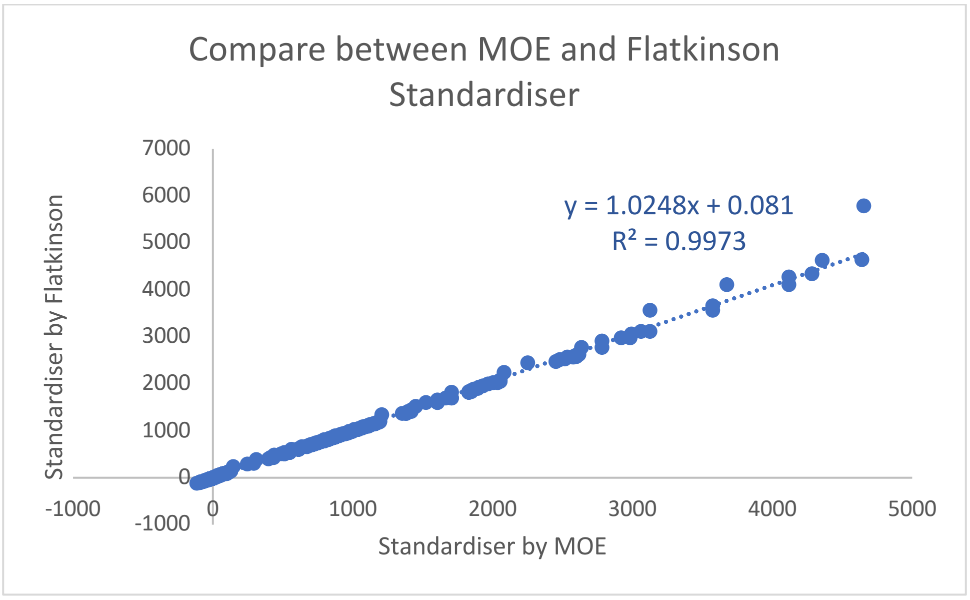


Figure 16 - Regressors representing the comparison between MOE and Bouysset Standardizers

Therefore, we chose to keep only the Flatkinson Standardiser and the descriptors extracted from it since the other Standardisers (MOE 2D and 3D) did not lead to relevant differences in the calculation of the descriptors.

4.4.2 STEP 1: Matching literature performance

The performance in Table 2, obtained by 51 Bouysset's descriptors, obtained shows the comparison between the Bouysset network and the networks formed using AB and GB. We can observe that the squared regression value and the root-mean-square error (RMSE) are shown to get an idea of the prediction quality and, at the same time, quantify the errors made. We can consider the values obtained as compared to those of Bouysset. Finally, Figure 17 (a.) and (b.) show the predictions with the AB and GB algorithms respectively. In Appendix 9, Appendix 10 and Appendix 11, you can see the complete metrics of the networks.

Table 2 – The performance of the test, validation and training set, and the error, in AB, GB and Bouysset networks are shown

Performance				
Model	R^2	RMSE	mean_train_score	mean_validation_score
AB	0.731	0.685	0.894	0.777
GB	0.713	0.709	0.974	0.796
Bouysset predictor	0.737	0.666	0.995	0.737

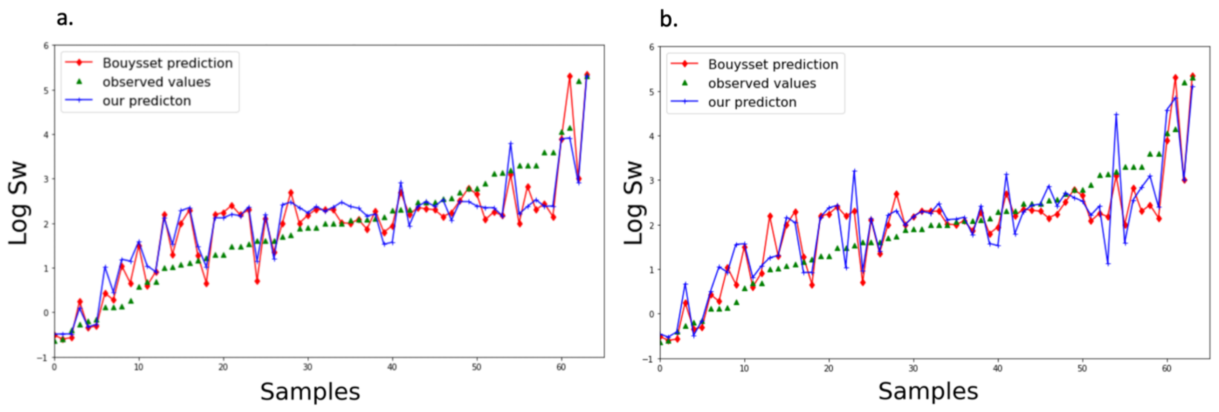


Figure 17 – Comparison of the results of the prediction models with the AB algorithm (a.) and the GB algorithm (b.). As the legend suggests, the red line represents the Bouysset trend, while the blue line represents the trend of our descriptor under consideration. The green dots represent the actual values.

4.4.3 STEP 2: Sweetness prediction using open-source and proprietary descriptors

Data preprocessing: removing less informative and correlated features

We performed the Bernoulli filter and then the cross-correlation from the obtained open-source and MOE descriptors, respectively 2102 and 422. At the end of the process, we got a number of features equal to 252 to avoid having a dataset that would bring the network into overfitting during the training.

The number of desired features was obtained by setting the threshold value of the cross-correlation in the two cases in the first is 0.92 while the second is 0.89. We see in Table 3 the results obtained from the single filtering passages.

Table 3 - Table shows the features selected during the individual filtering steps

Source descriptors	*Start Program*	*After Bernoulli*	*After Correlation*
OpenSource	2102	1684	252
MOE	422	295	252

Choice of the best feature selection method

We obtained three databases: one with the list of 51 descriptors used by Bouysset, the second with 422 descriptors from MOE, and the last with 2102 features from the open-source modules. These values were obtained by discarding descriptors whose value could not be extracted for each molecule.
Using the open-source features with the feature selection methods, we obtained the results in Table 4 that compare the selection methods applied using both AB and GB.

Table 4 - Number of features selected with AB and GB using RFE, GA, Mlxtend features selection methods and open-source descriptors

Regressor	*Source descriptors*	*After RFE*	After GA	After Mlxtend
AB	OpenSource	240	38	27
GB	OpenSource	200	33	30

The same steps were performed using the features selected by the MOE, and the Table 5 shows the results obtained using the 3 feature selections methods.

Table 5 - The number of features selected with AB and GB using RFE, GA, Mlxtend features selection methods and MOE descriptors

Regressor	*Source descriptors*	*After RFE*	After GA	After Mlxtend
AB	MOE	32	36	19
GB	MOE	34	29	21

Performance achieved with feature selection methods

we show the results of feature selection in Table 6

Table 6 - The performance obtained with AB and GB using RFE, GA, Mlxtend features selection methods and open-source descriptors

Regressor	*Source descriptors*	$R^2 score$ *RFE*	$R^2 score$ GA	$R^2 score$ Mlxtend
AB	OpenSource	0.715	0.667	0.724
GB	OpenSource	0.694	0.608	0.655

The same steps were performed using the features selected by the MOE, and the Table 7 shows the results obtained using the 3 feature selection methods.

Table 7 - The performance obtained with AB and GB using RFE, GA, Mlxtend features selection methods and MOE descriptors.

Regressor	*Source descriptors*	$R^2 score$ *RFE*	$R^2 score$ GA	$R^2 score$ Mlxtend
AB	MOE	0.695	0.731	0.711
GB	MOE	0.707	0.680	0.710

Looking at the results of using open-source descriptors, we can infer that the RFE algorithm provides good performance but uses too many features. The genetic algorithm provides fewer features and takes less time. However, the performance obtained is not among the best. Finally, although the Mlxtend algorithm requires a higher computational cost and takes more time, it selects fewer features, and the networks achieve the best performance. Using MOE descriptors, we obtained good results using all 3 feature selection methods. In this context, the GA algorithm provided the best results and a short time, while Mlxtend, compared to the others, continues to use fewer features. The performance obtained using the two different descriptors is comparable. Although the MOE descriptors received the best results, the use of proprietary software that is not freely accessible led us to opt for the open-source descriptors and the Mlxtend method of feature selection.

Optimisation of best models

From the choices made previously, we end up with the two models AB and GB with the following structures (Table 8)

Table 8 - Parameters of optimized models with AB and GB algorithms

Parameters of optimized models	
AB	GB
Flatkinson Standardiser	Flatkinson Standardiser
Mlxtend Features Selection	Mlxtend Features Selection
Open-source database with 27 descriptors	Open-source database with 30 descriptors
AB algorithm	GB algorithm

The list of 27 and 30 descriptors can be found in Appendix 8.

The results obtained with the two previously trained models are shown below. These models are composed of the AB and GB algorithm. The respective starting feature DBs were extracted using Mlxtend and obtained 27 and 30 features, respectively. Table 9 shows the relative performance obtained using the test, validation and training set. Finally, Figure 18(a.) and (b.) show the two predictors' trends compared with Bouysset.

Table 9 – The performance of the test, validation and training set, and the error, in AB, GB and Bouysset networks are shown

Performance				
Model	R^2	RMSE	mean_train_score	mean_validation_score
AB	0.737	0.673	0.897	0.847
GB	0.685	0.732	0.923	0.862
Bouysset predictor	0.737	0.666	0.995	0.737

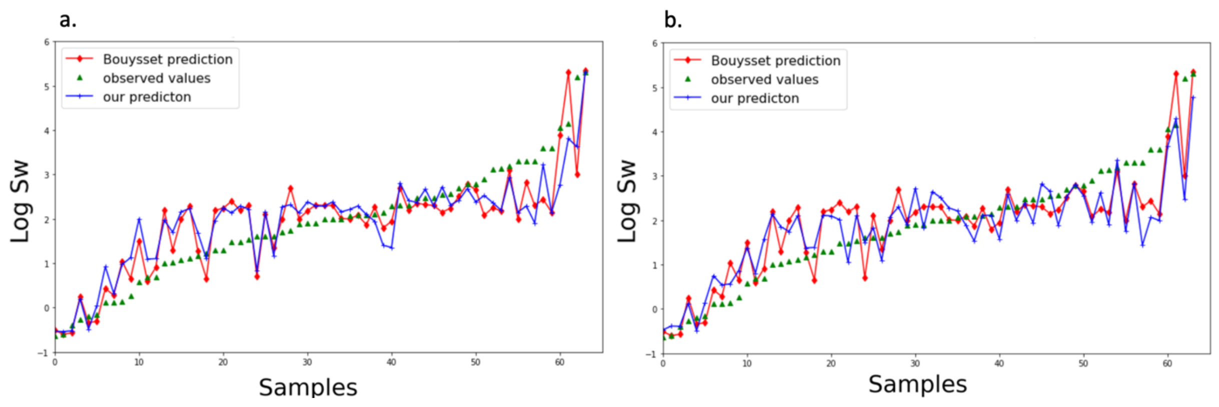


Figure 18 - Comparison of the results of the prediction models with the AB algorithm (a.) and the GB algorithm (b.). As the legend suggests, the red line represents the Bouysset trend, while the blue line represents the trend of our descriptor under consideration. The green dots represent the actual values.

Applicability and Reliability Domains

After optimizing the models, the top 10 features of each model were selected to calculate the applicability and reliability of the prediction. This selection was made by taking the features in descending order of importance. From which the applicability domain and its size were calculated.

Figure 19 (a.) shows the list of features selected from AB, while Figure 19 (b.) shows those obtained from GB.

Without going into too much detail, we would still like to point out the nature of the most important descriptors. In Figure 19, we can see that descriptors are calculated based on properties such as polarizability (AATS0p), the correlation between electronegativity values (GATS1se), mass (bcutm4), and topological charge (JGI4). We can see that all categories focus, albeit on different properties, on calculating the charges present in the molecular structure and the mass of the compound.

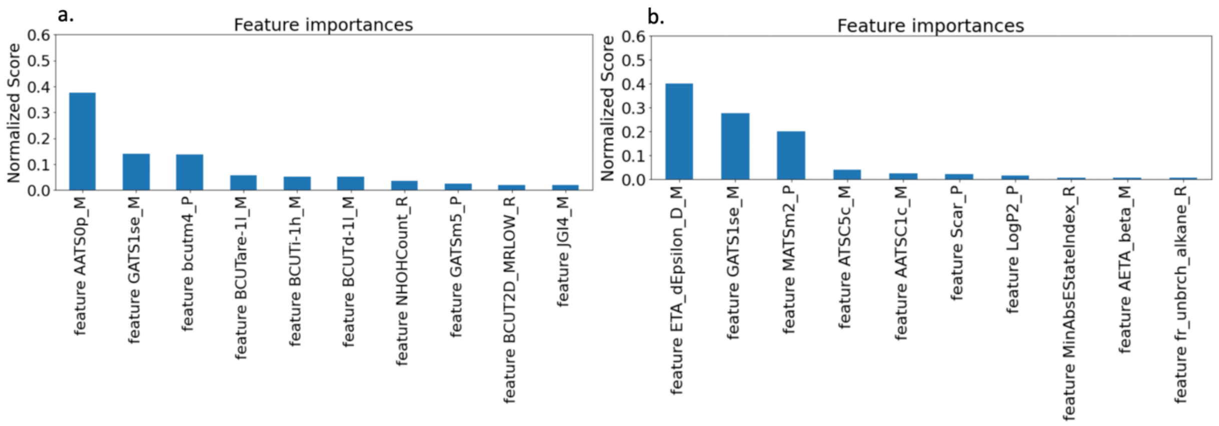


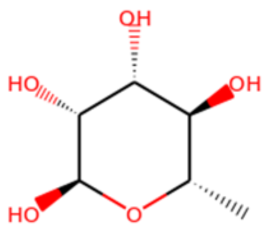
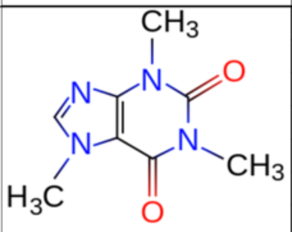
Figure 19 - List of features sorted by the importance of the AB (a.) and GB (b.).

After selecting the features with which to construct the domain, in Table 10, we show an example of an analysis performed using two molecules. The first row of the table shows the α-L-Rhamnopyranose molecule that belongs to the training domain, while the second row shows the caffeine molecule.

Looking at the results in the column of applicability, we can say that only in the first case the predictor is applicable. In other words, it provides an indication of possible use.

In the last column is examined the reliability, the function calculates the number of molecules that are less than the threshold distance (in this case 1.51) from the molecule of examination. For the first case, we find a normalized value of 0.143 to the number of molecules present in the train DB (252). This means that 36 molecules are present within the threshold distance. On the contrary, for caffeine not being inside the train DB, there are no molecules close to it.

Table 10 – It shows an example of two molecules on which the applicability domain is evaluated

Molecole	Structure	Applicability	Reliability
α-L-Rhamnopyranose		True	0.143
Caffeine		False	0

4.5 Discussion

From the first results obtained, we can infer that the two MOE Standardizers (2D and 3D) did not provide differences in the calculation of descriptors Figure 16 compared to Flatkinson's. However, the test was based on comparing the 51 descriptors mentioned in Bouysset's work [6]; it would be interesting to compare a more extensive set of descriptors, perhaps all available open-source ones.

Now, analyzing the performance of our source network compared to Bouysset's, which can be found in Table 2, we observe that the performance values are comparable to Bouysset's. Furthermore, Figure 17 shows that the AB and GB networks perform very similarly to Bouysset's predictor using the test set consisting of 64 molecules. Looking carefully at the graph, it can be seen for the GB network Figure 17 on the right, especially in the range from 30 to 40 it shows a much more similar performance to Bouysset's instead of the corresponding AB network. Despite this, it obtains a worse result. The reason is due to some predictions with high error, similar to spikes, for some molecules.

In Table 4 and Table 5 show a substantial difference in descriptor selection depending on the DB used (MOE or open-source). In particular, the RFE method starts with the same number of descriptors, 252 after the cross-correlation filter, but with the MOE DB, it selects fewer descriptors. Comparing the results, we have 240 and 200 with the open-source DB down to 32 and 34 with the MOE database. It can be observed that, albeit slightly, the performance (Table 6 and Table 7) when switching between the two DBs makes the AB network worse and the GB network better. This effect is also marked with the Mlxtend selection method. Moreover, we want to point out that even if this method has performances slightly lower than the RFE, it arrives to select 19 and 21 descriptors.

The exception is made by GA, where the decrease in AB's performance and increase in GB's performance is not there. When changing descriptors, we have a significant performance improvement, and slightly fewer descriptors are selected.

Turning to the results provided by optimized AB and GB, Table 9, where we used Mlxtend as the feature selection method and open-source descriptors (27 and 30) as DB, we see comparable performance to Bouysset. However, these models use fewer descriptors. Looking at Table 2 and Table 9, we can tell that our AB and GB networks have a minor performance degradation by moving from the train set to the validation set. This difference becomes even less pronounced in optimized AB and GB using the open-source descriptors, Table 9.

However, we can see in both Table 2 and Table 9 that the GB network tends to overfit more than the AB network. From the analysis in Figure 18, we do not notice a consistent deviation of the predicted values from the actual values. This leads us to infer that there is no Bias in our networks.

4.6 Conclusions

The idea behind this work was to create a predictor of sweetness molecules from the most common machine learning techniques currently adopted, as in the work of Wiener and colleagues [25].

In this work, we took a database of molecules whose sweetness was measured in the laboratory [8]. The molecules within it were subsequently standardized before extracting features. This process was made possible by the use of three Standardizers, which were then compared. Once the molecular descriptors were extracted, methods (RFE, GA, and Mlxtend) were used to choose some features and discard those that did not improve predictor performance.

After a parameter optimization, the created subset of features was used in our predictors (AB and GB). At the end of these steps, we obtained a model capable of evaluating the sweetness of a molecule under investigation. Additionally, using the domain definition functions, we assessed whether the predictor result is applicable and reliable.

Creating the final two models using AB and GB with open-source descriptors leads us to have excellent performance with fewer features than those used in other works such as Bouysset and Goel [6,30]. The results lead us to say that the feature selection phase currently has significant room for improvement.

As observed, the same models with MOE descriptors achieve even higher performance. Being the use of MOE only as a standardizer reductive and observing the work, for example of Kotsampasakou [35], we can think of an interesting development for the future, namely the application of the Flatkison standardizer and MOE in series to better standardize the starting DB.

An additional improvement for the future would be to have a more extensive database. The DB should also include molecules that are not sweet or have low sweetness. This would help the predictor, in addition to learning the level of sweetness from specific characteristics, penalize the result in the presence of properties that characterize non-sweet molecules (e.g., caffeine). Zheng also proposed this procedure in his work [36].

In conclusion, we are satisfied with the results obtained from this study, which aims to provide interesting insights and a solid starting point for the creation of a machine learning model that predicts sweet taste.

5 Appendix

Appendix 1 -Category descriptors Chemopy

Molecular descriptor category:
Molecular descriptors Constitutional descriptors
Topological descriptors
Connectivity descriptors
Kappa descriptors
Basak descriptors
E-state descriptors
Burden descriptors
Autocorrelation descriptors
Charge descriptors
Molecular property descriptors
MOE-type descriptors
Geometric descriptors
CPSA descriptors
WHIM descriptors
MoRSE descriptors
RDF descriptors

Appendix 2 - Category descriptors Rdkit

Descriptor Family
Gasteiger/Marsili Partial Charges
BalabanJ
BertzCT
Ipc
HallKierAlpha
Kappa1 - Kappa3
Phi
Chi0, Chi1
Chi0n - Chi4n
Chi0v - Chi4v
MolLogP
MolMR
MolWt
ExactMolWt
HeavyAtomCount
HeavyAtomMolWt
NHOHCount
NOCount
NumHAcceptors
NumHDonors
NumHeteroatoms
NumRotatableBonds
NumValenceElectrons
NumAmideBonds
Num{Aromatic,Saturated,Aliphatic}Rings
Num{Aromatic,Saturated,Aliphatic}{Hetero,Carbo}cycles
RingCount
FractionCSP3
NumSpiroAtoms
NumBridgeheadAtoms
TPSA
LabuteASA
PEOE_VSA1 - PEOE_VSA14
SMR_VSA1 - SMR_VSA10
SlogP_VSA1 - SlogP_VSA12
EState_VSA1 - EState_VSA11
VSA_EState1 - VSA_EState10
MQNs
Topliss fragments
Autocorr2D
BCUT2D

Appendix 3 - Category descriptors Mordred

Descriptor Family	
ABCIndex	InformationContent
AcidBase	KappaShapeIndex
AdjacencyMatrix	Lipinski
Aromatic	LogS
AtomCount	McGowanVolume
Autocorrelation	MoRSE
BCUT	MoeType
BalabanJ	MolecularDistanceEdge
BaryszMatrix	MolecularId
BertzCT	MomentOfInertia
BondCount	PBF
CPSA	PathCount
CarbonTypes	Polarizability
Chi	RingCount
Constitutional	RotatableBond
DetourMatrix	SLogP
DistanceMatrix	TopoPSA
EState	TopologicalCharge
EccentricConnectivityIndex	TopologicalIndex
ExtendedTopochemicalAtom	VdwVolumeABC
FragmentComplexity	VertexAdjacencyInformation
Framework	WalkCount
GeometricalIndex	Weight
GravitationalIndex	WienerIndex
HydrogenBond	ZagrebIndex

Appendix 4 - Category descriptors MOE

Descriptor Family
Physical Properties
Subdivided Surface Areas
Atom Counts and Bond Counts
Kier&Hall Connectivity and Kappa Shape Indices
Adjacency and Distance Matrix Descriptors
Pharmacophore Feature Descriptors
Partial Charge Descriptors
Surface Area, Volume and Shape Descriptors
MOPAC Descriptors
Conformation Dependent Charge Descriptors

Appendix 5 – List of 51 Bouysset descriptors

List of 51 Bouysset descriptors	
BertzCT	MATS1s
EState_VSA10	GATS1dv
HallKierAlpha	GATS1s
MaxAbsEStateIndex	GATS1se
MaxPartialCharge	GATS1p
MinEStateIndex	GATS2p
ATS0Z	GATS2i
AATS4d	BCUTc-1h
AATS0p	AXp-1d
AATS1p	AXp-2d
AATS5p	AETA_alpha
ATSC2c	ETA_dAlpha_B
ATSC3c	ETA_dEpsilon_D
ATSC1dv	ETA_psi_1
ATSC2s	AMID_O
AATSC2c	RotRatio
AATSC3c	GATSp2
AATSC1dv	IC1
AATSC2dv	MATSm2
AATSC2s	MATSm5
AATSC3s	MATSp2
AATSC1Z	bcute1
AATSC0v	bcute2
AATSC0p	bcutm2
AATSC0i	MATS1c
AATSC2i	

Appendix 6 – List of selected features from open-source database.

List of selected features					
‘SpMAD_A_M’	‘ATSC5s_M’	‘GATS1v_M’	‘AETA_beta_ns_d_M’	‘QHmin_P’	‘Hy_P’
‘VE1_A_M’	‘ATSC6s_M’	‘GATS3v_M’	‘AETA_eta_M’	‘Qomax_P’	‘SIC1_P’
‘nSpiro_M’	‘ATSC1v_M’	‘GATS4v_M’	‘AETA_eta_F_M’	‘Qcmax_P’	‘SIC2_P’
‘nBridgehead_M’	‘ATSC3v_M’	‘GATS5v_M’	‘ETA_eta_B_M’	‘Qhmax_P’	‘SIC3_P’
‘nS_M’	‘ATSC4v_M’	‘GATS1se_M’	‘AETA_eta_BR_M’	‘Chiv6ch_P’	‘IC1_P’
‘AATS0dv_M’	‘ATSC5v_M’	‘GATS2se_M’	‘ETA_dAlpha_A_M’	‘Aweight_P’	‘IC4_P’
‘AATS1dv_M’	‘ATSC7v_M’	‘GATS3se_M’	‘ETA_dAlpha_B_M’	‘Smax8_P’	‘bcutm5_P’
‘AATS2dv_M’	‘ATSC8v_M’	‘GATS1p_M’	‘ETA_epsilon_5_M’	‘Smax12_P’	‘bcutm4_P’
‘AATS3dv_M’	‘ATSC1se_M’	‘GATS3p_M’	‘ETA_dEpsilon_B_M’	‘Smax15_P’	‘bcutm1_P’
‘AATS4dv_M’	‘ATSC5se_M’	‘GATS2i_M’	‘ETA_dEpsilon_D_M’	‘Smax16_P’	‘bcute2_P’
‘AATS5dv_M’	‘ATSC6se_M’	‘GATS3i_M’	‘fMF_M’	‘Smax18_P’	‘MinAbsEStateIndex_R’
‘AATS4d_M’	‘ATSC0p_M’	‘BCUTc-1h_M’	‘IC1_M’	‘Smax33_P’	‘qed_R’
‘AATS5d_M’	‘ATSC3p_M’	‘BCUTc-1l_M’	‘BIC0_M’	‘Smin8_P’	‘NumRadicalElectrons_R’
‘AATS1s_M’	‘ATSC5p_M’	‘BCUTdv-1h_M’	‘MIC0_M’	‘Smin12_P’	‘FpDensityMorgan3_R’
‘AATS2s_M’	‘ATSC8p_M’	‘BCUTdv-1l_M’	‘MIC1_M’	‘Smin15_P’	‘BCUT2D_MWLOW_R’
‘AATS4s_M’	‘ATSC2i_M’	‘BCUTd-1h_M’	‘ZMIC3_M’	‘Smin16_P’	‘BCUT2D_LOGPLOW_R’
‘AATS5s_M’	‘ATSC3i_M’	‘BCUTd-1l_M’	‘Lipinski_M’	‘Smin18_P’	‘BCUT2D_MRHI_R’
‘AATS1v_M’	‘ATSC5i_M’	‘BCUTs-1h_M’	‘GhoseFilter_M’	‘Smin33_P’	‘BCUT2D_MRLOW_R’
‘AATS4se_M’	‘ATSC6i_M’	‘BCUTs-1l_M’	‘FilterItLogS_M’	‘Scar_P’	‘BalabanJ_R’
‘AATS5se_M’	‘ATSC8i_M’	‘BCUTZ-1h_M’	‘piPC1_M’	‘Shal_P’	‘HallKierAlpha_R’
‘AATS0p_M’	‘AATSC1c_M’	‘BCUTv-1l_M’	‘piPC10_M’	‘Save_P’	‘NHOHCount_R’
‘AATS4i_M’	‘AATSC0dv_M’	‘BCUTare-1l_M’	‘RotRatio_M’	‘DS_P’	‘fr_C_O_noCOO_R’
‘AATS5i_M’	‘AATSC0d_M’	‘BCUTi-1h_M’	‘JGI1_M’	‘GATSm1_P’	‘fr_N_O_R’
‘ATSC2c_M’	‘AATSC2s_M’	‘BCUTi-1l_M’	‘JGI2_M’	‘GATSm2_P’	‘fr_aldehyde_R’
‘ATSC3c_M’	‘AATSC0p_M’	‘C3SP2_M’	‘JGI3_M’	‘GATSm3_P’	‘fr_allylic_oxid_R’
‘ATSC4c_M’	‘AATSC5p_M’	‘C1SP3_M’	‘JGI4_M’	‘GATSm4_P’	‘fr_amide_R’
‘ATSC5c_M’	‘AATSC0i_M’	‘C3SP3_M’	‘JGI5_M’	‘GATSm5_P’	‘fr_aniline_R’
‘ATSC6c_M’	‘MATS1s_M’	‘Axp-2d_M’	‘JGI6_M’	‘GATSm8_P’	‘fr_aryl_methyl_R’
‘ATSC7c_M’	‘MATS5p_M’	‘Axp-3d_M’	‘JGI7_M’	‘GATSv1_P’	‘fr_bicyclic_R’
‘ATSC8c_M’	‘GATS2c_M’	‘Axp-1dv_M’	‘JGI8_M’	‘GATSe1_P’	‘fr_ester_R’
‘ATSC1dv_M’	‘GATS3c_M’	‘Axp-2dv_M’	‘JGI9_M’	‘GATSp3_P’	‘fr_ketone_R’
‘ATSC2dv_M’	‘GATS4c_M’	‘Axp-3dv_M’	‘JGI10_M’	‘GATSp4_P’	‘fr_ketone_Topliss_R’
‘ATSC3dv_M’	‘GATS5c_M’	‘MZ_M’	‘JGT10_M’	‘GATSp5_P’	‘fr_lactone_R’
‘ATSC4dv_M’	‘GATS2dv_M’	‘NsCH3_M’	‘TopoShapeIndex_M’	‘GATSp8_P’	‘fr_methoxy_R’
‘ATSC5dv_M’	‘GATS3dv_M’	‘NdCH2_M’	‘SRW05_M’	‘MATSm1_P’	‘fr_para_hydroxylation_R’
‘ATSC6dv_M’	‘GATS4dv_M’	‘NssCH2_M’	‘TSRW10_M’	‘MATSm2_P’	‘fr_sulfone_R’
‘ATSC7dv_M’	‘GATS1d_M’	‘NdsCH_M’	‘phi_P’	‘MATSm3_P’	‘fr_unbrch_alkane_R’
‘ATSC8dv_M’	‘GATS2d_M’	‘NddssS_M’	‘LDI_P’	‘MATSv1_P’	‘ATSm4_P’
‘ATSC2d_M’	‘GATS3d_M’	‘ETA_shape_p_M’	‘Mnc_P’	‘MATSe1_P’	‘Getov_P’
‘ATSC5d_M’	‘GATS4d_M’	‘ETA_shape_y_M’	‘Mpc_P’	‘MATSe3_P’	‘LogP2_P’
‘ATSC6d_M’	‘GATS5d_M’	‘ETA_shape_x_M’	‘QCss_P’	‘ATSC8d_M’	‘GATS5s_M’
‘ATSC7d_M’	‘GATS3s_M’	‘AETA_beta_M’	‘Qmax_P’	‘AETA_beta_s_M’	‘QCmin_P'

Appendix 7- List of selected features from MOE database.

List of selected features					
AM1_dipole'	'chi1v'	'lip_don'	'SlogP'	'vsurf_D2'	'vsurf_W1'
'ast_fraglike'	'chi1v_C'	'lip_druglike'	'SlogP_VSA0'	'vsurf_D3'	'vsurf_W2'
'ast_fraglike_ext'	'chi1_C'	'lip_violation'	'SlogP_VSA1'	'vsurf_D4'	'vsurf_W3'
'ast_violation'	'dens'	'logP(o/w)'	'SlogP_VSA2'	'vsurf_D5'	'vsurf_W4'
'ast_violation_ext'	'density'	'logS'	'SlogP_VSA3'	'vsurf_D6'	'vsurf_W5'
'a_acc'	'diameter'	'MNDO_dipole'	'SlogP_VSA4'	'vsurf_D7'	'vsurf_W6'
'a_base'	'E_ang'	'mutagenic'	'SlogP_VSA5'	'vsurf_D8'	'vsurf_Wp1'
'a_don'	'E_ele'	'opr_brigid'	'SlogP_VSA6'	'vsurf_DD12'	'vsurf_Wp2'
'a_donacc'	'E_oop'	'opr_leadlike'	'SlogP_VSA7'	'vsurf_DD13'	'vsurf_Wp3'
'a_hyd'	'E_sol'	'opr_nrot'	'SlogP_VSA8'	'vsurf_DD23'	'vsurf_Wp4'
'a_IC'	'E_str'	'opr_violation'	'SlogP_VSA9'	'vsurf_DW12'	'vsurf_Wp5'
'a_ICM'	'E_tor'	'PEOE_PC+'	'SMR'	'vsurf_DW13'	'vsurf_Wp6'
'a_nBr'	'FCharge'	'PEOE_PC-'	'SMR_VSA0'	'vsurf_DW23'	'vsurf_Wp7'
'a_nC'	'GCUT_PEOE_0'	'PEOE_RPC+'	'SMR_VSA1'	'vsurf_EDmin1'	'vsurf_Wp8'
'a_nCl'	'GCUT_PEOE_1'	'PEOE_RPC-'	'SMR_VSA2'	'vsurf_EDmin2'	'Weight'
'a_nF'	'GCUT_PEOE_2'	'PEOE_VSA+0'	'SMR_VSA3'	'vsurf_EDmin3'	'weinerPath'
'a_nH'	'GCUT_PEOE_3'	'PEOE_VSA+1'	'SMR_VSA4'	'vsurf_EWmin1'	'weinerPol'
'a_nI'	'GCUT_SLOGP_0'	'PEOE_VSA+2'	'SMR_VSA5'	'vsurf_EWmin2'	'b_single'
'a_nN'	'GCUT_SLOGP_1'	'PEOE_VSA+3'	'SMR_VSA6'	'vsurf_EWmin3'	'b_triple'
'a_nO'	'GCUT_SLOGP_2'	'PEOE_VSA+4'	'SMR_VSA7'	'vsurf_G'	'chi0'
'a_nS'	'GCUT_SLOGP_3'	'PEOE_VSA+5'	'std_dim1'	'vsurf_HB1'	'chi0v'
'balabanJ'	'GCUT_SMR_0'	'PEOE_VSA+6'	'std_dim2'	'vsurf_HB2'	'chi0v_C'
'BCUT_PEOE_0'	'GCUT_SMR_1'	'PEOE_VSA-0'	'std_dim3'	'vsurf_HB3'	'vsurf_HB4'
'BCUT_PEOE_1'	'GCUT_SMR_2'	'PEOE_VSA-1'	'TPSA'	'vsurf_ID1'	'vsurf_HB5'
'BCUT_PEOE_2'	'GCUT_SMR_3'	'PEOE_VSA-2'	'VAdjEq'	'vsurf_ID2'	'vsurf_HB6'
'BCUT_PEOE_3'	'glob'	'PEOE_VSA-3'	'VAdjMa'	'vsurf_ID3'	'vsurf_HL1'
'BCUT_SLOGP_0'	'h_ema'	'PEOE_VSA-4'	'VDistEq'	'vsurf_ID4'	'vsurf_HL2'
'BCUT_SLOGP_1'	'h_emd'	'PEOE_VSA-5'	'VDistMa'	'vsurf_ID5'	'vsurf_A'
'BCUT_SLOGP_2'	'h_emd_C'	'PEOE_VSA-6'	'vol'	'vsurf_ID6'	'vsurf_CP'
'BCUT_SLOGP_3'	'h_logD'	'PEOE_VSA_FPNEG'	'VSA'	'vsurf_ID7'	'vsurf_CW1'
'BCUT_SMR_0'	'h_logP'	'PEOE_VSA_FPPOS'	'vsa_acc'	'vsurf_ID8'	'vsurf_CW2'
'BCUT_SMR_1'	'h_logS'	'PEOE_VSA_HYD'	'vsa_base'	'vsurf_IW1'	'vsurf_CW3'
'BCUT_SMR_2'	'h_log_dbo'	'PEOE_VSA_NEG'	'vsa_don'	'vsurf_IW2'	'vsurf_IW6'
'BCUT_SMR_3'	'h_log_pbo'	'PEOE_VSA_PNEG'	'vsa_hyd'	'vsurf_IW3'	'vsurf_IW7'
'bpol'	'h_pavgQ'	'PEOE_VSA_POL'	'vsa_other'	'vsurf_IW4'	'vsurf_IW8'
'b_1rotN'	'h_pKa'	'PEOE_VSA_POS'	'vsa_pol'	'vsurf_IW5'	'vsurf_R'
'b_1rotR'	'h_pKb'	'PEOE_VSA_PPOS'	'KierA2'	'vsurf_CW4'	'vsurf_S'
'b_double'	'h_pstates'	'petitjean'	'pmi1'	'vsurf_CW5'	'vsurf_V'
'b_max1len'	'h_pstrain'	'petitjeanSC'	'pmiY'	'vsurf_CW6'	'vsurf_D1'
'b_rotN'	'Kier1'	'PM3_dipole'	'pmiZ'	'vsurf_CW7'	'reactive'
'b_rotR'	'Kier2'	'KierFlex'	'radius'	'vsurf_CW8'	'rgyr'
'KierA1'	'Kier3'	'lip_acc'	'rsynth'	'chi0_C'	'KierA3'

Appendix 8 - List of 27 and 30 descriptor selections with I and II model

Model I (with AB)	Model II (with GB)
AATS0p_M	MATSm2_P
GATS1se_M	ETA_dEpsilon_D_M
bcutm4_P	GATS1se_M
BCUTare-1l_M	ATSC5c_M
BCUTi-1h_M	AATSC1c_M
BCUTd-1l_M	Scar_P
NHOHCount_R	LogP2_P
GATSm5_P	MinAbsEStateIndex_R
BCUT2D_MRLOW_R	AETA_beta_M
JGI4_M	Smin16_P
ATSC8d_M	fr_unbrch_alkane_R
QCmin_P	C3SP3_M
GATS5c_M	NHOHCount_R
fr_amide_R	fr_N_O_R
BCUTc-1h_M	fr_para_hydroxylation_R
ATSC6d_M	GhoseFilter_M
AATS1dv_M	fr_aniline_R
ETA_eta_B_M	fr_sulfone_R
QCmax_P	fr_lactone_R
QHmax_P	fr_ketone_Topliss_R
ATSC2d_M	fr_ketone_R
Smax18_P	fr_aryl_methyl_R
BCUTs-1h_M	fr_aldehyde_R
GATS5d_M	fr_allylic_oxid_R
AETA_beta_M	Lipinski_M
GhoseFilter_M	NumRadicalElectrons_R
NdsCH_M	nBridgehead_M
nSpiro_M	NdsCH_M
NddssS_M	

Appendix 9 – AB prediction performance

```
Performance with StandardScaler
R: 0.855
R^2: 0.731
|R^2-R_0_elev2|/R^2: 0.210
k: 0.981
R_0_elev2: 0.577
|R^2-R_0_primo_elev2|/R^2: 0.011
k_primo: 0.928
R_0_primo_elev2: 0.722
RMSE: 0.685
MAE: 0.568
```

Appendix 10 - GB prediction performance

```
Performance with StandardScaler
R: 0.855
R^2: 0.731
|R^2-R_0_elev2|/R^2: 0.210
k: 0.981
R_0_elev2: 0.577
|R^2-R_0_primo_elev2|/R^2: 0.011
k_primo: 0.928
R_0_primo_elev2: 0.722
RMSE: 0.685
MAE: 0.568
```

Appendix 11 – Bouysset prediction performance

```
Performance of Bouysset predict
R: 0.858
R^2: 0.737
|R^2-R_0_elev2|/R^2: 0.082
k: 0.989
R_0_elev2: 0.660
|R^2-R_0_primo_elev2|/R^2: 0.015
k_primo: 0.925
R_0_primo_elev2: 0.7369
RMSE: 0.666
MAE: 0.496
```

6 Acknowledgements

First of all, I would like to thank Professor Marco Agostino Deriu for the opportunity to do this thesis work on machine learning. I would also like to thank my co-advisors, Marta Malavolta and Lorenzo Pallante, for their constant support and guidance during the writing of this work.

The research has been developed as part of the VIRTUOUS project, funded by the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie—RISE Grant Agreement No 872181.

Also, a big thank you goes to my family and my girlfriend for accompanying me on this educational path during these years.

Thank you to my friends for listening and for all the lighthearted moments.

Juan Antonio di Lorenzo

7 REFERENCES

1. Pallante L, Malavolta M, Grasso G, et al. On the human taste perception: Molecular-level understanding empowered by computational methods. *Trends in Food Science and Technology*. 2021;116:445-459. doi:10.1016/j.tifs.2021.07.013

2. Temussi P. The history of sweet taste: Not exactly a piece of cake. *Journal of Molecular Recognition*. 2006;19(3). doi:10.1002/jmr.767

3. Belloir C, Neiers F, Briand L. Sweeteners and sweetness enhancers. *Current Opinion in Clinical Nutrition and Metabolic Care*. 2017;20(4). doi:10.1097/MCO.0000000000000377

4. Poirier N, Roudnitzky N, Brockhoff A, et al. Efficient production and characterization of the sweet-tasting brazzein secreted by the yeast pichia pastoris. *Journal of Agricultural and Food Chemistry*. 2012;60(39). doi:10.1021/jf301600m

5. Thermofisher. Fluo4-AM spectrum. https://www.thermofisher.com/order/catalog/product/F14201.

6. Bouysset C, Belloir C, Antonczak S, Briand L, Fiorucci S. Novel scaffold of natural compound eliciting sweet taste revealed by machine learning. *Food Chemistry*. Published online 2020. doi:10.1016/j.foodchem.2020.126864

7. SHALLENBERGER RS. Hydrogen Bonding and the Varying Sweetness of the Sugars. *Journal of Food Science*. 1963;28(5). doi:10.1111/j.1365-2621.1963.tb00247.x

8. Chéron JB, Casciuc I, Golebiowski J, Antonczak S, Fiorucci S. Sweetness prediction of natural compounds. *Food Chemistry*. Published online 2017. doi:10.1016/j.foodchem.2016.10.145

9. Ruiz-Aceituno L, Hernandez-Hernandez O, Kolida S, Moreno FJ, Methven L. Sweetness and sensory properties of commercial and novel oligosaccharides of prebiotic potential. *LWT*. Published online 2018. doi:10.1016/j.lwt.2018.07.038

10. Ahmed J, Preissner S, Dunkel M, Worth CL, Eckert A, Preissner R. SuperSweet-A resource on natural and artificial sweetening agents. *Nucleic Acids Research*. Published online 2011. doi:10.1093/nar/gkq917

11. Cao DS, Xu QS, Hu QN, Liang YZ. ChemoPy: Freely available python package for computational biology and chemoinformatics. *Bioinformatics*. 2013;29(8). doi:10.1093/bioinformatics/btt105

12. Landrum G. RDKit : A software suite for cheminformatics , computational chemistry , and predictive modeling. *Components*. Published online 2011.

13. Moriwaki H, Tian YS, Kawashita N, Takagi T. Mordred: A molecular descriptor calculator. *Journal of Cheminformatics*. 2018;10(1). doi:10.1186/s13321-018-0258-y

14. Anaconda Inc. Anaconda Distribution. *Anaconda*. Published online 2019.

15. Anderson E, Veith GD, Weininger D. SMILES: a line notation and computerized interpreter for chemical structures. In: ; 1987.

16. Ratsch G, Onoda T, Muller KR. An improvement of AdaBoost to avoid overfitting. *Advances in Neutral Information Processing Systems, Kitakyushu, Japan*. Published online 1998.

17. Schapire RE. The strength of weak learnability. *Machine Learning*. 1990;5(2). doi:10.1007/bf00116037

18. Friedman JH. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*. 2001;29(5). doi:10.1214/aos/1013203451

19. Prachi Prakash. AdaBoost and Gradient Boost – Comparitive Study Between 2 Popular Ensemble Model Techniques. Published October 27, 2020. Accessed November 29, 2021. https://www.analyticsvidhya.com/blog/2020/10/adaboost-and-gradient-boost-comparitive-study-between-2-popular-ensemble-model-techniques/

20. Golbraikh A, Tropsha A. Beware of q2! In: *Journal of Molecular Graphics and Modelling*. Vol 20. ; 2002:269-276. doi:10.1016/S1093-3263(01)00123-1

21. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*. 2011;12.

22. Raschka S. MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. *Journal of Open Source Software*. 2018;3(24). doi:10.21105/joss.00638

23. Conrad M. Hans Joachim Bremermann 1926–1996. *Biosystems*. 1996;39(1). doi:10.1016/s0303-2647(96)90013-1

24. Genetic Programming and Emergent Intelligence. In: *Advances in Genetic Programming*. ; 2020. doi:10.7551/mitpress/1108.003.0009

25. Dagan-Wiener A, Nissim I, ben Abu N, Borgonovo G, Bassoli A, Niv MY. Bitter or not? BitterPredict, a tool for predicting taste from chemical structure. *Scientific Reports*. Published online 2017. doi:10.1038/s41598-017-12359-7

26. Tuwani R, Wadhwa S, Bagler G. BitterSweet: Building machine learning models for predicting the bitter and sweet taste of small molecules. *Scientific Reports*. Published online 2019. doi:10.1038/s41598-019-43664-y

27. Hanser T, Barber C, Marchaland JF, Werner S. Applicability domain: towards a more formal definition. *SAR and QSAR in environmental research*. 2016;27(11). doi:10.1080/1062936X.2016.1250229

28. Barber CB, Dobkin DP, Huhdanpaa H. The Quickhull Algorithm for Convex Hulls. *ACM Transactions on Mathematical Software*. 1996;22(4). doi:10.1145/235815.235821

29. Huang W, Shen Q, Su X, et al. BitterX: A tool for understanding bitter taste in humans. *Scientific Reports*. Published online 2016. doi:10.1038/srep23450

30. Goel A, Gajula K, Gupta R, Rai B. In-silico prediction of sweetness using structure-activity relationship models. *Food Chemistry*. Published online 2018. doi:10.1016/j.foodchem.2018.01.111

31. Sigoillot M, Laffitte A, Neiers F, Briand L. Sweet taste inhibitors: Therapeutic propects. *Cahiers de Nutrition et de Diététique*. 2015;50(5). doi:10.1016/j.cnd.2015.02.003

32. Zhong M, Chong Y, Nie X, Yan A, Yuan Q. Prediction of sweetness by multilinear regression analysis and support vector machine. *Journal of Food Science*. Published online 2013. doi:10.1111/1750-3841.12199

33. Chemical Computing Group ULC. Molecular Operating Environment (MOE), 2019.01 User Guide. *Scientific Computing & Instrumentation*. Published online 2019.

34. Francis Atkison. Flatkinson Standardiser. Published online 2018.

35. Eleni Kotsampasakou. Predicting liver toxicity on basis of transporter interaction profiles. Published online 2016.

36. Zheng S, Chang W, Xu W, Xu Y, Lin F. e-Sweet: A machine-learning based platform for the prediction of sweetener and its relative sweetness. *Frontiers in Chemistry*. Published online 2019. doi:10.3389/fchem.2019.00035