FINAL MASTER'S PROJECT

Master degree in Chemical Engineering

MODELIZATION AND VALIDATION OF ACUTE TOXICITY ON AQUATIC FAUNA APPLYING CLASSIFICATION AND REGRESSION METHODS.

Author:	Marc Garcia Campos
Director:	Samir Kanaan Izquierdo
Date:	June 2021

Resum

L’objectiu principal d’aquest projecte es el d’aplicar diferents mètodes de classificació i de regressió per a poder predir, en un futur, la toxicitat d’una substància química sobre una espècie de peix, *Pimephales promelas*, a partir de la seva estructura química. La finalitat és poder evitar la mort i el patiment d’aquests animals utilitzats per a les avaluacions toxicològiques Per això, s’ha obtingut una base de dades de 908 molècules amb la seva toxicitat sobre el peix d’estudi i, a través de l’empresa KODE Chemoinformatics s’han obtingut aproximadament 4000 indicadors moleculars per a cada substància els quals s’han reduït a 2500 a través d’un filtratge. Un cop obtinguda la base de dades i amb ajuda del programa Python, s’ha obtingut diferents combinacions de variables que s’han testat amb mètodes de Machine Learning de regressió y classificació aconseguint un error mitjà absolut normalitzat d’un 7,3% com a mínim a partir dels mètodes de regressió aplicats i amb els de classificació una precisió del 60%. Tot i l’obtenció d’uns resultats acceptables per regressió, no són prou fiables per a substituir el mètode tradicional d’avaluació de la toxicitat d’un químic ja que un error de càlcul podria esdevenir en resultats fatals.

Resumen

El objetivo principal de este proyecto es el de aplicar diferentes métodos de clasificación y de regresión para poder predecir, en un futuro, la toxicidad de una sustancia química sobre una especie de pescado, *Pimephales promelas*, a partir de su estructura química. La finalidad es poder evitar la muerte y el sufrimiento de estos animales utilizados para las evaluaciones toxicológicas Por ello, se ha obtenido una base de datos de 908 moléculas con su toxicidad sobre el pescado de estudio y, a través de la empresa KODE Chemoinformatics se han obtenido aproximadamente 4000 indicadores moleculares para cada sustancia los cuales se han reducido a 2500 a través de un filtrado. Una vez obtenida la base de datos y con ayuda del programa Python, se ha obtenido diferentes combinaciones de variables que se han testado con métodos de Machine Learning de regresión y clasificación consiguiendo un error medio absoluto normalizado de un 7,3% como mínimo a partir de los métodos de regresión aplicados y con los de clasificación una precisión del 60%. Aunque la obtención de unos resultados aceptables por regresión, no son lo suficientemente fiables para sustituir el método tradicional de evaluación de la toxicidad de un químico ya que un error de cálculo podría convertirse en resultados fatales.

Abstract

The main objective of this project is to apply different classification and regression methods to be able to predict, in the future, the toxicity of a chemical substance on a specific fish, *Pimephales promelas*, based on its chemical structure. The purpose is to be able to avoid the death and suffering of these animals used for toxicological evaluations. Therefore, a database of 908 molecules with their toxicity on the study fish has been obtained and, through the company KODE chemoinformatics, it has been obtained approximately 4000 molecular indicators for each substance which have been reduced to 2500 through filtering. Once the database has been obtained and with the help of the program Python, different combinations of variables have been obtained that have been tested with Machine Learning methods of regression and classification, obtaining a normalized mean absolute error of a minimum of 7,3% from the Regression methods applied and with those of classification a precision of 60%. Although obtaining acceptable results through regression, they are not reliable enough to replace the traditional method of evaluating the toxicity of a chemical since a calculation error could turn into fatal results.

Acknowledgments

Alberto Manganaro, CEO and CTO of KODE Chemoinformatics must be mentioned for his unselfish help and support, providing the molecular descriptors from one of their chemoinformatic programs, DRAGON 7.0

Index

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

1. Introduction

One of the most important properties of a chemical along with the toxicity on humans, is the toxicity to aquatic organisms. A chemical can't be placed in the market without demonstrating that a chemical is safer for human health and environment since REACH regulation entered into force in June 2007. One goal if this regulation is the avoidance of unnecessary testing, because measuring the toxicity of a chemical implies the death of a high number of animals or not the death but an extreme suffering being this ethically questionable.

To obtain this, companies that want to supply the same chemical in Europe are required to share toxicological test in the registration process avoiding the repetition them unnecessarily among other methodologies like in vitro and in silico test methods.

To improve this, a new approach has emerged, the quantitative structure-activity relationships (QSAR) analysis. QSAR models are regression and classification models that are used in the biological and chemical science based on the prediction of a biological activity (i.e., the toxicity on a fish) from physico-chemical properties or theorical molecular descriptors.

This study is focused on the development of a QSAR models in order to predict the toxicity of new chemicals towards the *Pimephales promelas* through regression and classification methods and comparing them looking for the best results. The models are developed with a data set of 908 organic molecules, being the output, the toxicity defined as LC_{50} 96 hours (Lethal concentration that kills 50% of the test animals during 96h) and approximately 4000 molecular descriptors as the attributes.

1.1. Objectives

- Design, training and evaluation of different machine learning classification models to predict the toxicity of a chemical towards the fathead minnow
- Design, training and evaluation of different machine learning regression models to predict the toxicity of fathead minnow
- Evaluate the applicability of these methods in the real world
-

1.2. Project planning

The project has been divided in 5 steps that can be seen at the figure above:

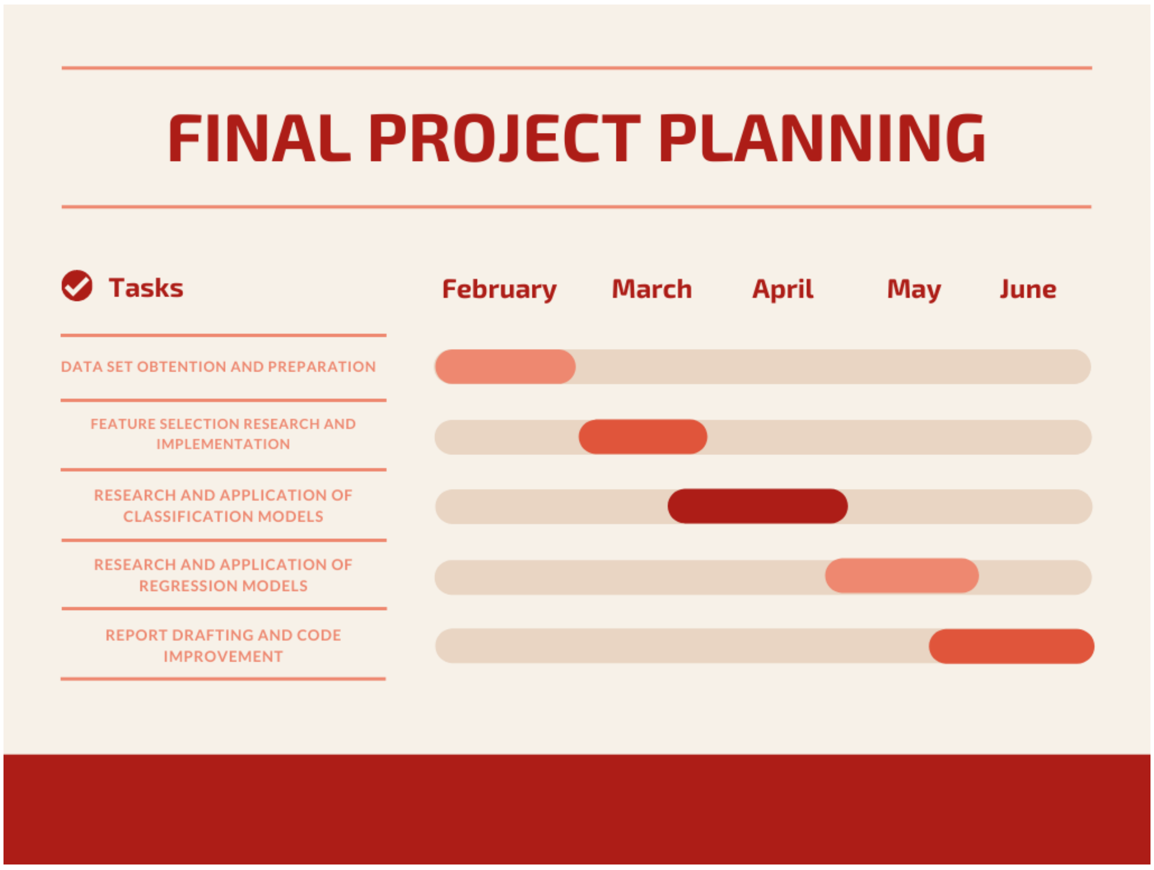


Figure 1. Gantt's diagram

1.3. The fish of study: *Pimephales promelas*

Figure 2. Fathead minnow (Pimephales promelas)(Source: Free 3D).

The fathead minnow is a temperate freshwater fish which is located in North America and Canada. It has been used for toxicity tests since the 1863 but it was not until the 1950s that this fish began to be widely used for regulatory ecotoxicology. They are being used from 48h -96h assays (short term) to partial and full life-cycle to determine the lethality of chemicals. They also are used for research applications focused in the development of QSAR models.

These fish are used widely nowadays because they short life cycle (5-6 months) in comparison with other species used before the 1950 resulting in more efficient assays of partial-full life cycle. Another reason why the popularity of these fish increased over the years is the tolerance to a wide range of water types and a well-defined reproductive cycle, making them a suitable species for cultivation in laboratories. [1]

1.4. Machine Learning

Machine learning studies computer algorithms that are improved automatically through the addition of data (training data). With this train different models based on the sample data can be obtained to make predictions or to take decisions. There exist 3 different approaches depending on the nature of the feedback, **Supervised learning, Unsupervised learning and Reinforcement learning.** This project is based on several machine learning models that are explained in detail below

1.4.1. Reinforcement learning

This first approach works on through trial and error and on a reward basis. To exemplify this, if the machine doesn't do the prediction well, its reward is equal to 0 or negative but if it does the correct action, it has positive rewards obtaining at the end a good solution. Doing this we can obtain an intelligent system that does good decision if obviously it has a good training. [2]

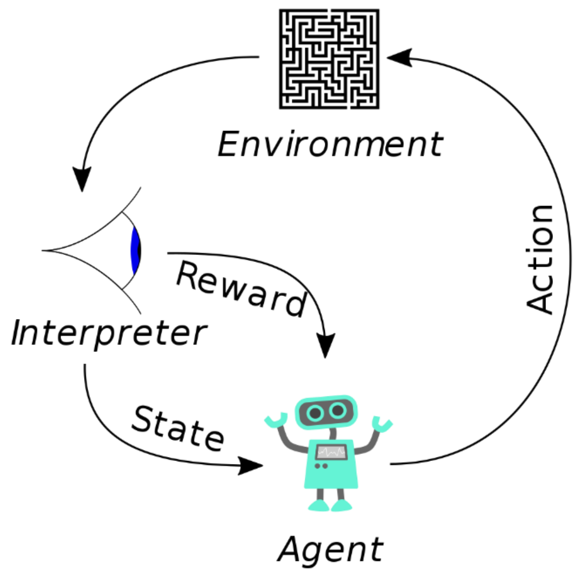


Figure 3. Reinforcement Learning Process. (Source: Megajuice)

1.4.2. Unsupervised learning

This technique uses only input and non-labelled data, this means that the machine must be able to find the structure of the data instead of responding to a feedback. This type of learning is commonly used to reduce the dimensionality of the data reducing the loss of information. From a whole set of characteristics, unsupervised learning is able to find a number of groups with similar characteristics defined by the data analyst

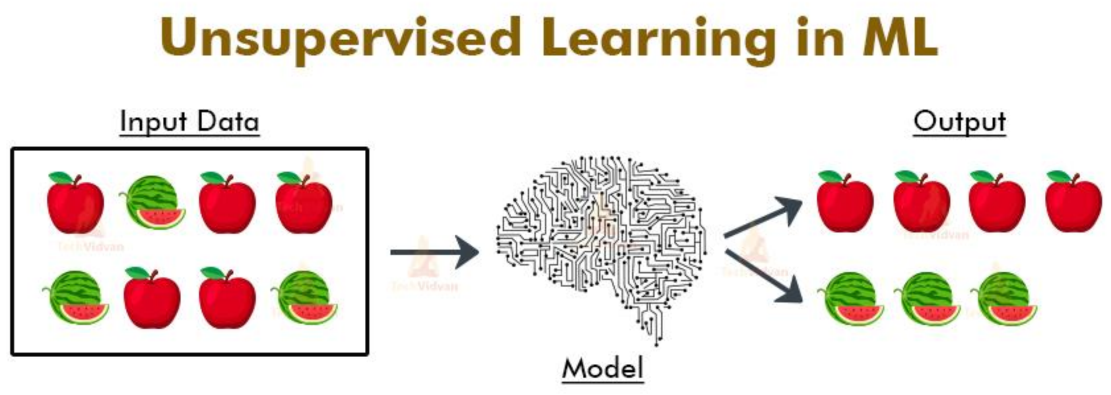


Figure 4. Unsupervised learning example. (Source: Techvidvan)

1.4.3. Supervised learning

This project is based on the supervised learning, a technique to deduce a function from a given training data. This training data consist on the input data (the characteristics) and the desired results with the given characteristics. If the output of the function is a numerical value, we are in a regression problem and regression methods must be applied, and if the output data is a class label it is a classification problem.

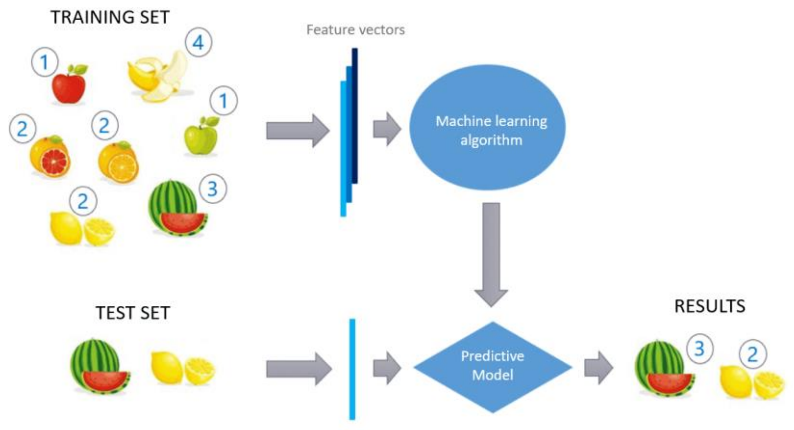


Figure 5. Supervised learning example (Source: Diego Calvo)

In this case, regression and classification methods have been applied where the desired result is the acute toxicity of a chemical substance on *Pimephales promelas* (in numerical and categorical value) and the input data is a list of characteristics of each substance.

The following regression and classification method are the most widely used and their fundamentals are very different from each other, making them interesting to compare in this project.

- Decision trees
- Ensemble methods (Bagging, bootstrapping, gradient boost, Random Forest, AdaBoost)
- K-NN
- Naïve Bayes
- Artificial neural networks
- Support vector machine

1.5. Supervised learning methods

1.5.1. k-NN

The method consists of finding a number k of training samples closest in distance to the new point (test) and predicting the label from these. Since it is a non-parametric method (it makes no assumptions about the distribution of the data), it is successful in data sets where the decision boundary is irregular.

To apply this method, it is usually necessary to normalize the attributes so that those with higher rank do not have more weight than the others. K-NN can be used in classification and regression done [3]

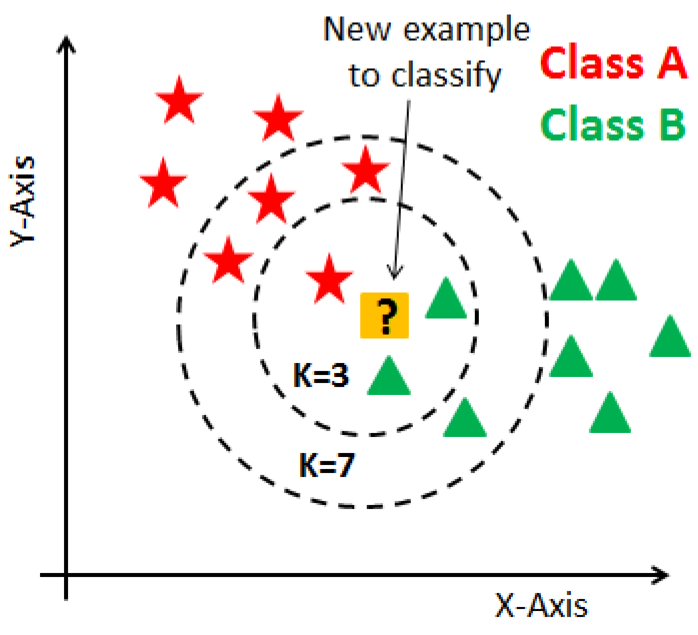


Figure 6. Example of k-NN method. Is k=2, the test point belongs to class B, if it is k=7, to class A. (Source: Datacamp)

This method first calculates the distance between the point to be classified and the rest of the items in the training dataset. Then, it selects the k closest points and the more times repeated class of the k closest points is the final class of the test point.

For the distance calculation, the most typical is the Euclidean distance.

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{p} (x_{ri} - x_{rj})^2}$$

Figure 7. Euclidean distance equation

This method has some limitations:

- It is a slow algorithm, since it must store all the training data.
- It is very sensitive to noise.
- It depends excessively on the distance between samples and the definition of this is not always adequate (there are different types of distances).
- It is very sensitive to irrelevant attributes and suffers greatly from the curse of dimensionality (need for a large number of samples as the variables increase).

The good point of this method is that is a very simple method to implement.

1.5.2. Decision tree

Decision trees are a supervised (the actual class is known) non-parametric (it makes no assumptions about the distribution followed by the data) method that consists of predicting the value of a target variable by learning simple rules deduced from the characteristics of the data. Each branch of a tree corresponds to a rule that decides between subsets of values of an attribute (internal) or makes a prediction of the class (terminal). Due to its nature, it is a simple algorithm to understand and interpret, requires little data preparation and handles all types of variables (numerical and categorical). [4]

Some limitations that this method presents are:

- Overfitting: creation of overly complex trees that do not generalize the data well. Can be solved by setting the minimum number of samples required for each grouping.
- Unstable to small data variations.
- It is very sensitive to the difference in size between classes in the training set (it is excessively influenced by the larger groups).
- High computational cost.

And some advantages that decision trees have are:

- It's a very easy method to understand.
- The data does not need an exhaustive pre-processing like other techniques.
- It can manage categorical and numerical data.
- The models obtained are robust.

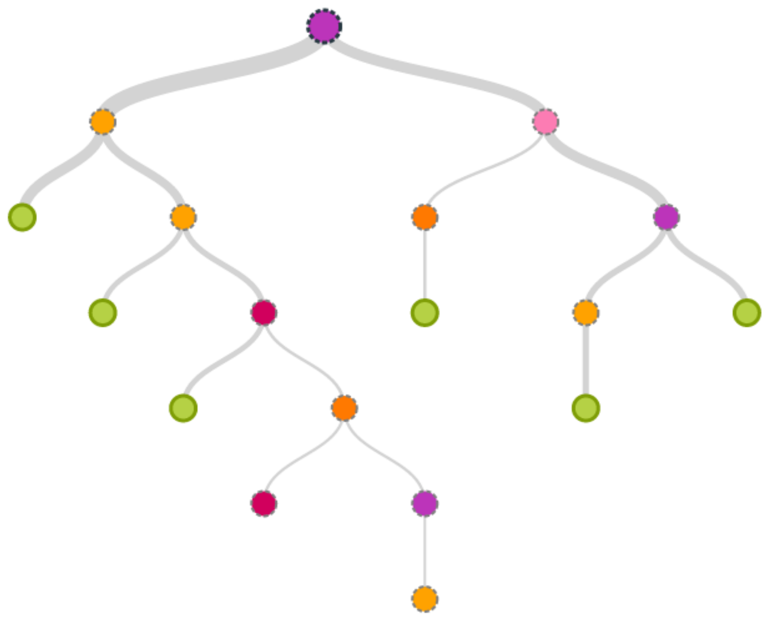


Figure 8. Decision Tree (Source: BigML)

1.5.3. Ensemble methods

Ensemble methods are techniques based in the combination of several base models with the objective of producing an optimal predictive model. There are two differentiated groups:

- Boosting methods: Estimators are built in a sequential mode and one tries to reduce the bias of the final combined estimator, some methods are AdaBoost and Gradient Boost among others.
- Averaging methods: Several estimators are built with independence and then, to the average of the predictions. This average usually Is better than a single estimator. Bagging and Random forest are averaging methods.

1.5.3.1. AdaBoost

This algorithm is based on the linear combination of rules from several weak classifiers to obtain a robust classifier with a low error, thus forming a high-performance classifier. These rules are created by assigning weights on the training set that are updated according to the results obtained:

in case of a correct classification, the weight is lowered. In case of a wrong classification, it raises it. Once this has been done, it runs the same tree again with the new weight value. This causes a weight proportional to the learning difficulty to be assigned to the observations of the training set.

This is a simple algorithm, with few parameters to set, and with an easily understandable interpretation of the rules adopted, being able to distinguish errors in the labelling thanks to the final weights adopted by the method. [5]

Some limitations of this method are:

- High computational cost
- Presents problems for data sets with more than two classes.

And some advantages:

- Easy to implement
- Low generalization error
- Works with a high range of classifiers

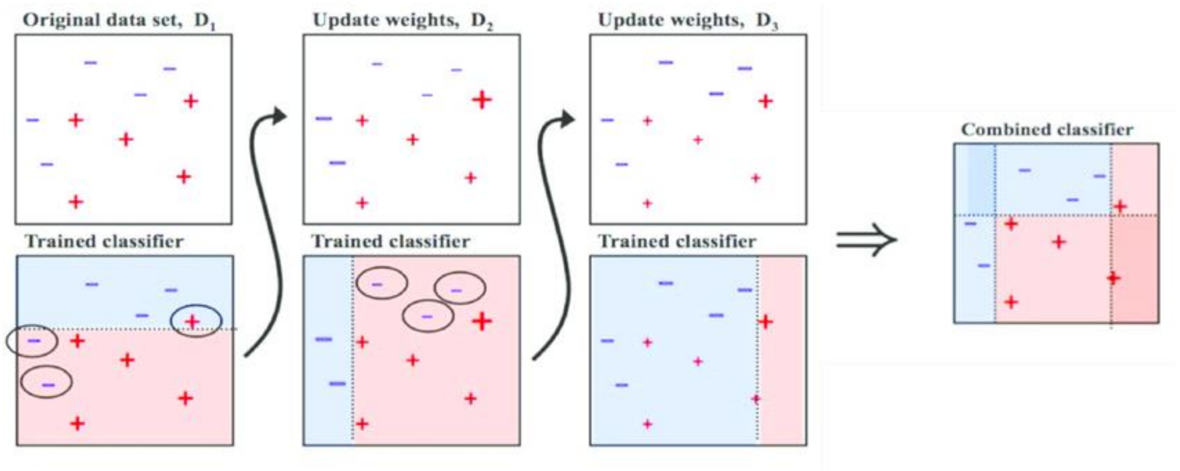


Figure 9. AdaBoost Classifier (Source: Towards Data Science)

1.5.3.2. Gradient Boost

A Gradient Boosting model consists of a set of individual decision trees, trained sequentially, so that each new tree tries to improve the errors of the previous trees. The prediction of a new observation is obtained by aggregating the predictions of all the individual trees that make up the model.

Many predictive methods generate global models in which a single equation applies to the entire sample space. When the use case involves multiple predictors, which interact with each other in a complex and nonlinear way, it is very difficult to find a single global model that is able to reflect the relationship between the variables. Tree-based statistical and machine learning methods encompass a set of nonparametric supervised techniques that manage to segment the space of predictors into simple regions, within which it is easier to handle interactions. It is this feature that gives them much of their potential.

Some limitations of this method: [6]

- When combining multiple trees, the interpretability of single-tree models is lost.
- When dealing with continuous predictors, they lose some of their information when categorizing them at the time of splitting the nodes.
- They are not able to extrapolate outside the range of predictors observed in the training data.

And some advantages:

- They are able to select predictors automatically.
- They can be applied to regression and classification problems.
- Since these are non-parametric methods, no specific distribution is required.
- In general, they require much less data cleaning and pre-processing compared to other statistical learning methods (i.e., they do not require standardization).
- They are not greatly influenced by outliers.
- If for any observation, the value of a predictor is not available, despite not being able to reach any terminal node, a prediction can be achieved using all observations belonging to the last node reached. The accuracy of the prediction will be reduced but at least it can be obtained.
- They have good scalability; they can be applied to data sets with a large number of observations.

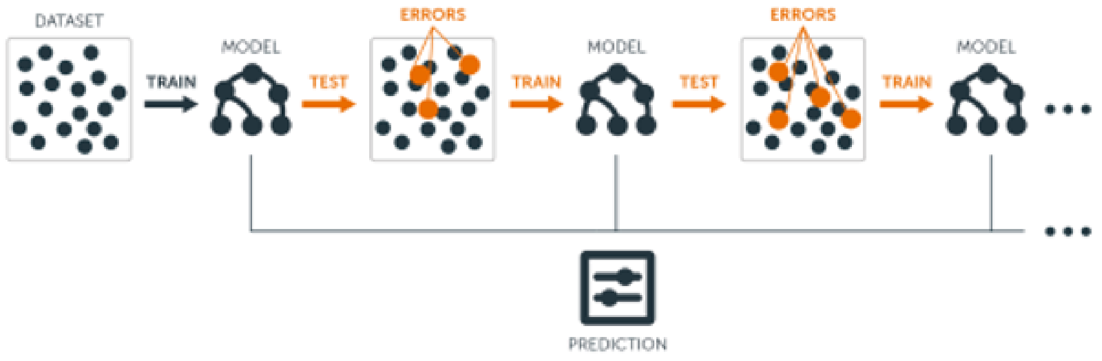


Figure 10. Gradient Boosting procedure (Source: Gradient Docs)

1.5.3.3. Bootstrapping

This method is a resample method that can be used to estimate the skill of several machine learning models. This method works building samples by drawing observations from a data sample and then, returning them to the data sample after they have been chosen. To asses the skill of a model, it is trained on the sample evaluating the performance of those samples that are not included in the sample.

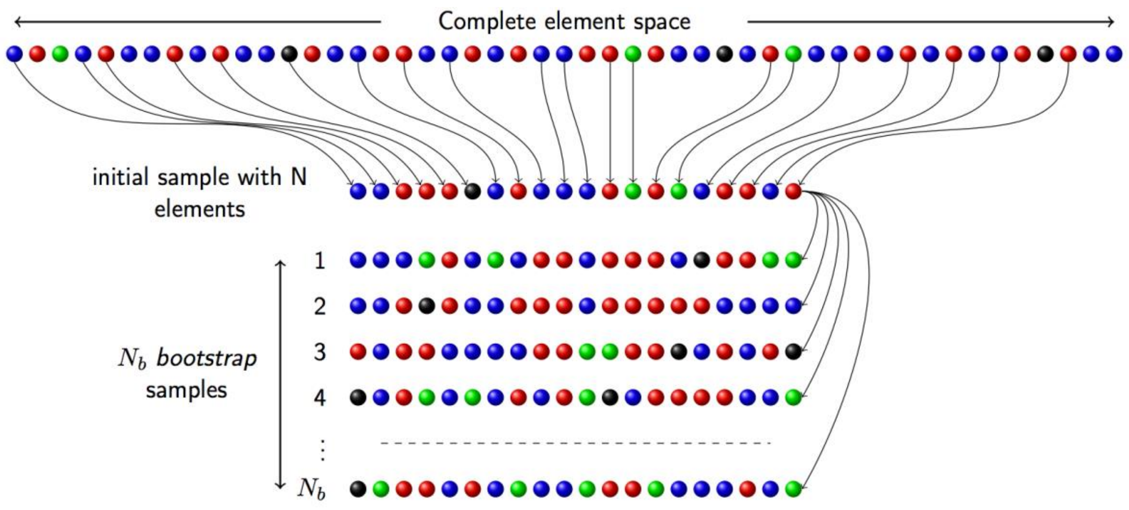


Figure 11. Bootsrapping resampling method (Source: Let the machines learn)

1.5.3.4. Bagging

Bagging is an ensemble meta-estimator that it can be created fitting multiple versions of a base estimator training it with modified training data set. This automatic learning is designed to improve the stability and the accuracy of other methods used in either classification and regression, generally used in decision trees. [7]

Some limitations of this method are:

- The resulting model can experience many biases when the proper procedure is ignored.
- Although bagging is highly accurate, it can be computationally expensive
- It introduces a loss of interpretability
- If there is no variance in the data set, Bagging technique does not significantly improve the model. It is recommended in models with high instability (data set with a lot of variance).

And some advantages:

- Decreases the variance of a data set by resampling with replacement
- Overfitting or overtraining of models is reduced. This is because the models cannot overlearn or memorize since none of them has all the training data.
- Improves prediction, since what is not detected by one model is detected by the others

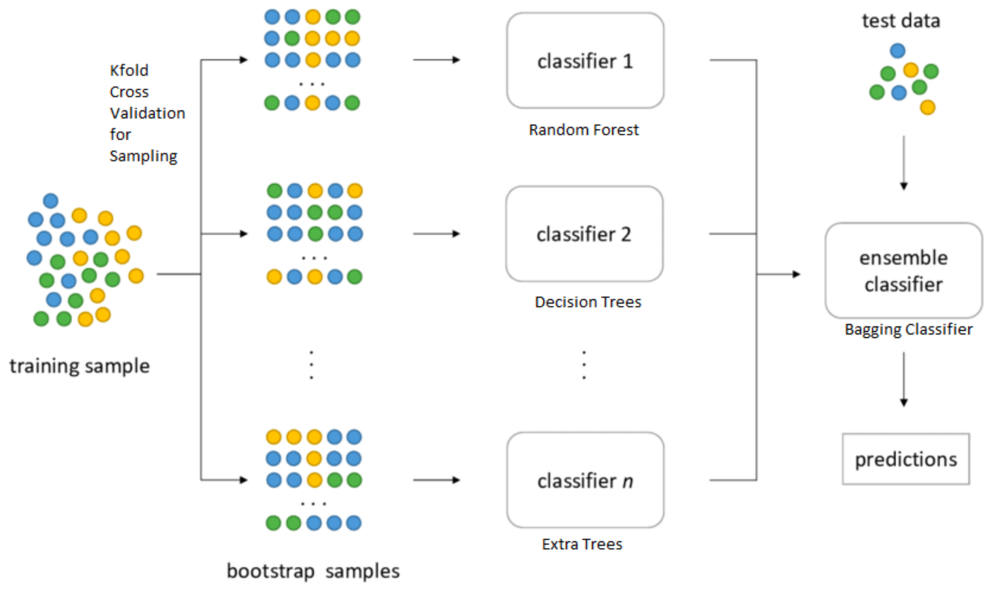


Figure 12. Bagging procedure (Source: Vitalflux.com)

1.5.3.5. Random forest

A Random Forest model consists of a set of individual decision trees, each trained with a random sample drawn from the original training data by bootstrapping. This implies that each tree is trained on slightly different data. In each individual tree, observations are distributed by bifurcations generating the tree structure until a terminal node is reached. The prediction of a new observation is obtained by aggregating the predictions of all the individual trees that make up the model. [8]

Random Forest is based on the bagging algorithm and uses the ensemble learning technique. It creates as many trees in the subset of data and combines the output of all the trees.

Some limitations of the Random Forest:

- Little control over what the model does.
- The training period is increased, the Random Forest requires much more time to train compared to the normal decision trees, since it generates many trees and makes the decision on most of the votes.
- Increased complexity of the model, many more weak trees are needed. This algorithm requires much more computational power and resources.
- Loss of interpretation

And some advantages:

- Is often robust to outliers and can handle them automatically.
- Can handle up to thousands of input variables and identify the most significant ones.
- Incorporates effective methods for estimating missing values
- There are very few assumptions and therefore data preparation is minimal
- Reduces the problem of overfitting in decision trees, also reduces variation and, as a consequence, improves accuracy

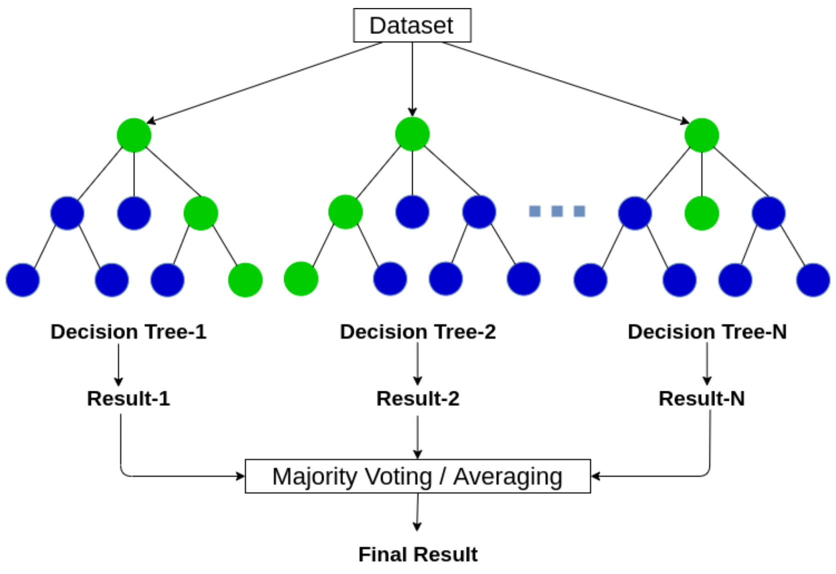


Figure 13. RandomForest Procedure (Source: Analytics Vidhya)

1.5.4. Naïve Bayes

It is a classifier based on a probabilistic model based on Bayes' theorem, which incorporates simplifying assumptions that can be summarized in the assumption of "naive" independence between variables or characteristics.

The algorithm classifies the new samples by assigning them a class that maximizes the conditional probability of the class in relation to the sequence observed in the attributes. This probability is estimated from the relative frequencies (maximum likelihood estimation) of the training set. The difference between the different Naive Bayes classifiers (Gaussian, Multinomial, Bernoulli) is the assumption made regarding the distribution of the features of the training set. [9]

There are three Naive Bayes algorithms which, as mentioned above, differ in the distribution assumed to the training set:

- **Gaussian Naive Bayes**: assumes a Gaussian distribution of characteristics.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Figure 14. Equation for a Gaussian Naive Bayes (Source: Towardsdatascience).

- **Multinomial Naive Bayes**: assumes a multinomial distribution (generalization of the binomial), which are estimated by maximum likelihood smoothing (relative frequency counting)

$$P(x_i \mid y) = \begin{cases} \dfrac{n!}{x_1! \cdots x_k!} p_1^{x_1} \cdots p_k^{x_k}, & \text{cuando } \sum_{i=1}^{k} x_i = n \\ 0 & \text{En otros casos,} \end{cases}$$

Figure 15. Equation for Multinomial Naive Bayes

- **Bernoulli Naive Bayes**: assumes a Bernoulli distribution (each variable has binary values).

Limitations of this method:

- Assumes independence between the attributes representing each sample.
- It is very sensitive to the difference in size between classes in the training set (it is excessively influenced by the larger groups).
- Although they are quite good classifiers, Naive Bayes algorithms are known to be poor estimators. Therefore, the probabilities obtained should not be taken too seriously.
- The Naive independence assumption will most likely not reflect what the data are like in the real world.
- When the test data set has a feature that has not been observed in the training set, the model will assign a probability of zero to it and it will be useless to make predictions. One of the main methods to avoid this is the smoothing technique, Laplace estimation being one of the most popular.

Advantages of Naïve Bayes:

- An easy and fast way to predict classes, for binary and multiclass classification problems.
- In cases where an independence assumption is appropriate, the algorithm performs better than other classification models, even with less training data.
- The decoupling of class-conditional feature distributions means that each distribution can be estimated independently as if it had only one dimension. This helps with problems arising from dimensionality and improves performance.

1.5.5. Support vector machines

The support vector machine (SVM) is another method for solving classification and regression problems.

The idea is to select a separation hyperplane that is equidistant from the closest examples of each class in order to achieve what is called a maximum margin on each side of the hyperplane. In addition, when defining the hyperplane, only training examples that are distant from the hyperplane by the margin distance are considered. These examples are called support vectors.

Some limitations of this method are:

- If the number of features is much larger than the number of samples, it is important to avoid over-fitting when choosing Kernel features and the regularization term.
- SVMs do not directly provide probability estimates, these are calculated using 5-fold cross-validation.

And the advantages of SVM are:

- Effective in spaces of large dimensions.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Effective in cases where the number of dimensions is greater than the number of samples.
- Different kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

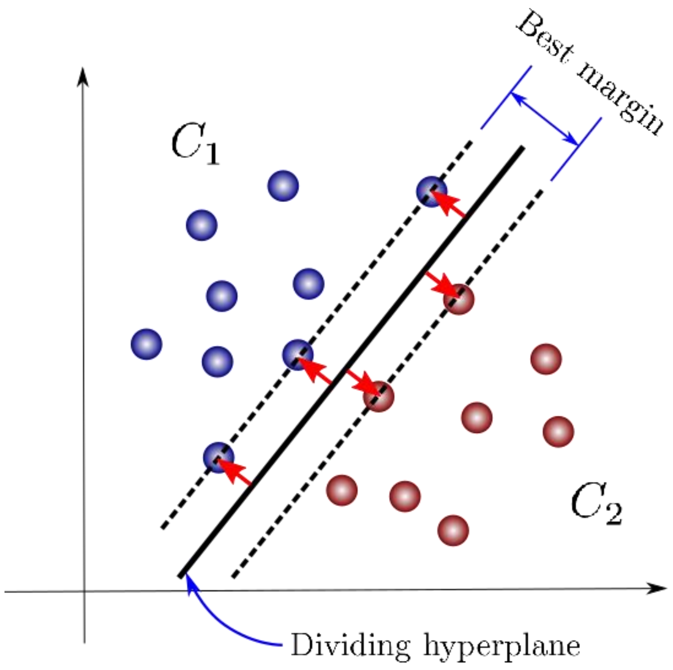


Figure 16. Support Vector Machine Example (Source: Towards Data Science)

1.5.6. Artificial Neural Network

An ANN is a structure composed of a number of interconnected units (artificial neurons), each unit possesses an input/output characteristic and implements a local computation or function. The output of any unit is determined by its interconnection with other units, and possibly by its internal units. The network usually develops a functionality usually through one or more forms.

It is therefore a massive arrangement of simple processing elements called neurons, which possess a high degree of interconnectivity between their elements, in which information can flow in cascade or backward. These arrangements are inspired by the biological nature of neurons.

These systems learn and train themselves, rather than being explicitly programmed, and excel in areas where solution or feature detection is difficult to express with conventional programming. To perform this machine learning, typically, an attempt is made to minimize a loss function that evaluates the network as a whole. The values of the neuron weights are updated in an attempt to reduce the value of the loss function. This process is performed by means of backward propagation.

The goal of the neural network is to solve problems in the same way as the human brain, although neural networks are more abstract. [10]

Some limitations of this method:

- Complexity of learning for large tasks, the more things a network needs to learn, the more complicated it will be to teach it.
- High learning time. This depends on two factors: first, if the number of patterns to be identified or classified increases, and second, if greater flexibility or adaptability of the neural network is required to recognize patterns that are extremely similar, more time must be invested in getting the network to converge to values of weights that represent what is to be taught.
- It does not allow interpreting what has been learned, the network by itself provides an output, a number, which cannot be interpreted by itself, but requires the intervention of the programmer and the application itself to find a meaning to the output provided.
- High amount of data for training, the more flexible the neural network is required to be, the more information it will have to be taught to perform the identification properly.

And finally, some advantages:

- ANNs have the ability to learn through a stage called the learning stage. This consists of providing the ANN with data as input while indicating the expected output (response).
- An ANN creates its own representation of the information inside it, relieving the user of this.
- Because an ANN stores information redundantly, it can continue to respond acceptably even if it is partially damaged.
- An ANN can handle unimportant changes in the input information, such as noisy signals or other changes in the input (e.g. if the input information is the image of an object, the corresponding response remains unchanged if the image changes a little in brightness or the object changes slightly).
- The structure of an ANN is parallel, so if this is implemented with computers or in special electronic devices, real-time responses can be obtained. [11]

1.6. Molecular descriptors used

By definition, *"The molecular descriptor is the final result of a logic and mathematical procedure which transforms chemical information encoded within a symbolic representation of a molecule into a useful number or the result of some standardized experiment."* [12]. These molecular descriptors are fundamental in some fields such as pharmaceutical, chemistry, etc. because they can be transformed from a “real body” to numbers allowing through this process their mathematical treatment of the chemical information.

Molecular descriptors can be clearly separated in two groups, the first one is the ones obtained through experimental measurements such as polarizability, molar refractivity among others. The second group are the ones obtained through theorical and can be calculated using chemoinformatic software, they can be simple descriptors obtained by counting some atom types or more complex derived from algorithms applied to a topological representation or descriptors derived from the geometrical representation of the molecular structure. [13]

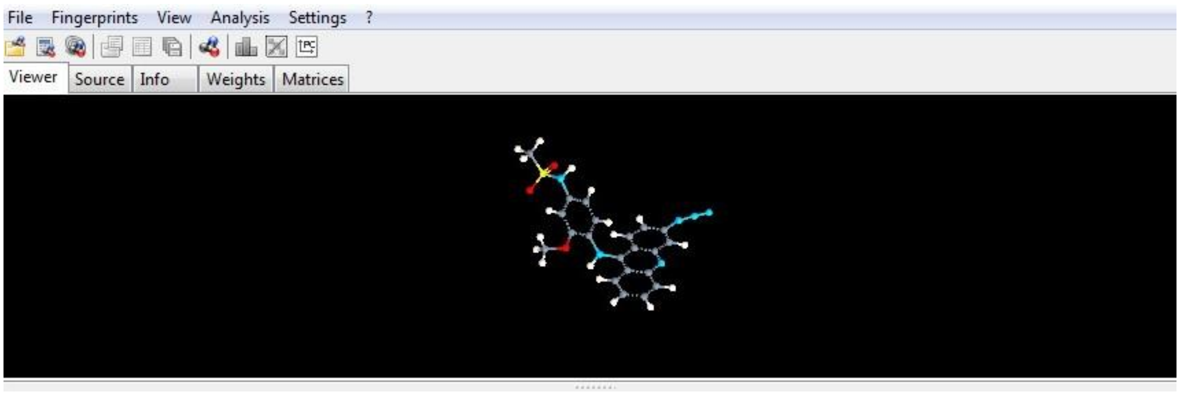


Figure 17. DRAGON 7.0 Software (Source: KODE Cheminformatics)

The list of 5270 molecular descriptors that KODE Cheminformatics provided is organized in 30 logical blocks for an easier management.

Table 1.Table 1. Logical blocks of the molecular descriptors

Constitutional

Ring descriptors
Topological indices
Walk and path counts
Connectivity indices
Information indices
2D matrix-based descriptors
2D autocorrelations
Burden eigenvalues
P-VSA-like descriptors
ETA indices
Edge adjacency indices
Geometrical descriptors
3D matrix-based descriptors
3D autocorrelations
RDF descriptors
3D-MoRSE descriptors
WHIM descriptors
GETAWAY descriptors
Randic molecular profiles

Functional groups count
Atom-centred fragments
Atom-type E-state indices
CATS 2D
2D Atom Pairs
3D Atom Pairs
Charge descriptors
Molecular properties
Drug-like indices
CATS 3D

Although the high number of molecular descriptors (5270) assessed in this theorical section are briefly explained the ones that are more relevant and used at the application of the models. These molecular descriptors are:

HyWi_B(m), *hyper-Wiener-like index (log function) from Burden matrix weighted by mass*

$$\mathrm{HyWi\_M}(w) = \ln\left\{1 + \frac{1}{2} \cdot \sum_{i=1}^{\mathrm{nSK}} \sum_{j=i}^{\mathrm{nSK}} \left(\left[\mathbf{M}(w)\right]_{ij}^{2} + \left[\mathbf{M}(w)\right]_{ij} \right) \right\}$$

Figure 18. Formula to obtain HyWi_B(m). (Source: KODE Chemoinformatics)

It is a topological attribute that is obtained after the application of a ser of basic algebraic operators to different graph-theorical matrices. In this case, the matrix used is a Burden matrix which is an augmented adjacency matrix derived from a H-depleted molecular graph.

SM1_B(m), *spectral moment of order 1 from Burden matrix weighted by mass*

$$\mathrm{SM1\_M}(w) = \mathrm{sgn}\left(\sum_{i=1}^{nSK} \lambda_i\right) \cdot \ln\left(1 + \left|\sum_{i=1}^{nSK} \lambda_i\right|\right)$$

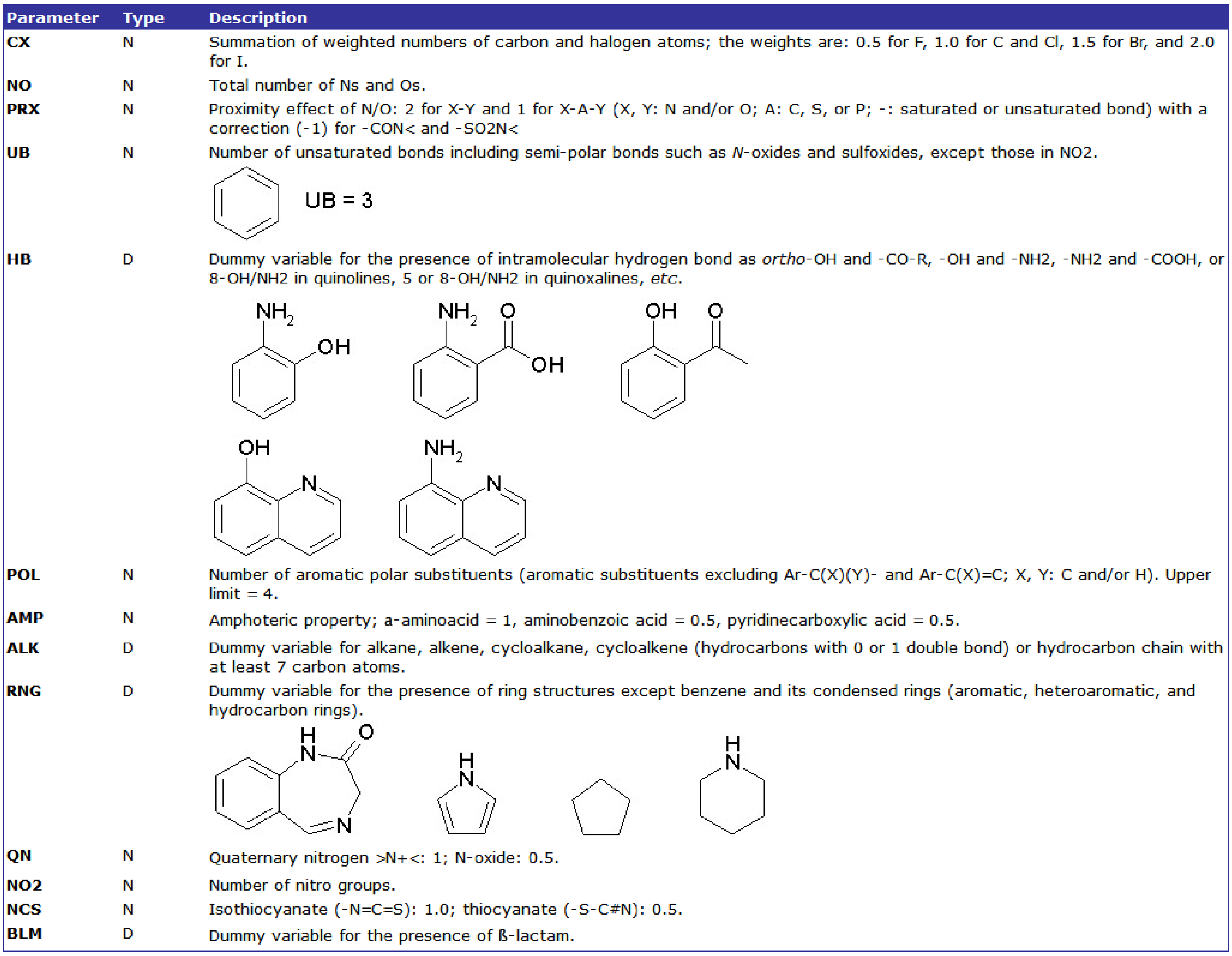
Figure 19. Formula to obtain SM1_B(m). (Source: KODE Chemoinformatics)

SM1_B(m) also belongs to the 2D matrix-based descriptor and it is a topological index based in the Burden matrix.

MLOGP, *Moriguchi octanol-water partition coeff. (logP)*: This value is a molecular property which is obtained from the Moriguchi logP model, based on a regression equation of 13 structural parameters [14]. The equation is:

$$\mathrm{MlogP} = -1.041 + 1.244(CX)^{0.6} - 1.017(NO)^{0.9} + 0.406(PRX) - 0.145(UB)^{0.8} + 0.511(HB) + 0.268(POL) - 2.215(AMP) + 0.912(ALK) - 0.392(RNG) - 3.684(QN) + 0.474(NO2) + 1.582(NCS) + 0.773(BLM)$$

Figure 20. MLOGP Equation (Source: KODE Chemoinformatics)



Parameter	Type	Description
CX	N	Summation of weighted numbers of carbon and halogen atoms; the weights are: 0.5 for F, 1.0 for C and Cl, 1.5 for Br, and 2.0 for I.
NO	N	Total number of Ns and Os.
PRX	N	Proximity effect of N/O: 2 for X-Y and 1 for X-A-Y (X, Y: N and/or O; A: C, S, or P; -: saturated or unsaturated bond) with a correction (-1) for -CON< and -SO2N<
UB	N	Number of unsaturated bonds including semi-polar bonds such as *N*-oxides and sulfoxides, except those in NO2. UB = 3
HB	D	Dummy variable for the presence of intramolecular hydrogen bond as *ortho*-OH and -CO-R, -OH and -NH2, -NH2 and -COOH, or 8-OH/NH2 in quinolines, 5 or 8-OH/NH2 in quinoxalines, *etc*.
POL	N	Number of aromatic polar substituents (aromatic substituents excluding Ar-C(X)(Y)- and Ar-C(X)=C; X, Y: C and/or H). Upper limit = 4.
AMP	N	Amphoteric property; a-aminoacid = 1, aminobenzoic acid = 0.5, pyridinecarboxylic acid = 0.5.
ALK	D	Dummy variable for alkane, alkene, cycloalkane, cycloalkene (hydrocarbons with 0 or 1 double bond) or hydrocarbon chain with at least 7 carbon atoms.
RNG	D	Dummy variable for the presence of ring structures except benzene and its condensed rings (aromatic, heteroaromatic, and hydrocarbon rings).
QN	N	Quaternary nitrogen >N+<: 1; N-oxide: 0.5.
NO2	N	Number of nitro groups.
NCS	N	Isothiocyanate (-N=C=S): 1.0; thiocyanate (-S-C#N): 0.5.
BLM	D	Dummy variable for the presence of ß-lactam.

Figure 21. Description of each parameter of MlogP equation (Source: KODE Chemoinformatics)

MLOGP2, *squared Moriguchi octanol-water partition coeff. (logP^2)*

ALOGP, *Ghose-Crippen octanol-water partition coeff. (logP)*: ALOGP is a molecular property calculated from a model that consist of a regression equation based on the hydrophobicity contribution of 115 atom types [15]. Each atom in every structure is classified into one of the 115 atom types. Then, estimated logP for any compound is given by the following formula:

$$\mathrm{AlogP} = \sum_i n_i a_i$$

Figure 22. Equation for ALOGP Calulation. (Source: KODE Chemoinformatics)

In the equation above, ni is the number of atoms of type i and ai is the corresponding hydrophobicity constant.

ALOGP2, *squared Ghose-Crippen octanol-water partition coeff. (logP^2)*

BLTF96, *Verhaar Fish base-line toxicity from MLOGP (mmol/l):* This attribute is based on Moriguchi LogP and is defined by Verhaar.

$$BLTF96 = -0.85 * MLogP - 1.39 \quad \text{(Verhaar Fish)}$$

Figure 23. Equation to obtain BLTF96 (Source: KODE Chemoinformatics)

BLTD48, *Verhaar Daphnia base-line toxicity from MLOGP (mmol/l):* This attribute is based on Moriguchi LogP and is defined by Verhaar.

$$BLTD48 = -0.95 * MLogP - 1.32 \quad \text{(Verhaar Daphnia)}$$

Figure 24. Equation to obtain BLTD48 (Source: KODE Chemoinformatics)

BLTA96, *Verhaar Algae base-line toxicity from MLOGP (mmol/l):* This attribute is based on Moriguchi LogP and is defined by Verhaar.

$$BLTA96 = -1.00 * MLogP - 1.23 \quad \text{(Verhaar Algae)}$$

Figure 25. Equation to obtain BLTA96 (Source: KODE Chemoinformatics)

H_Dz(m), *Harary-like index from Barysz matrix weighted by mass*

$$H\_M(w) = \sum_{i=1}^{nSK} \left[M(w) \right]_{ii} + \sum_{i=1}^{nSK-1} \sum_{j=i+1}^{nSK} \frac{1}{\left[M(w) \right]_{ij}}$$

Figure 26. Equation to obtain H_Dz(m). (Source: KODE Chemoinformatics)

This attribute is a 2D matrix-based descriptor based on Barysz matrices which are distance matrices accounting contemporarily for the presence of heteroatoms and multiple bonds in the molecule.

P_VSA_ppp_D, *P_VSA-like on potential pharmacophore points, D - hydrogen-bond donor:* P_VSA_ppp_D descriptor is based on atom assignment to the most common Potential Pharmacophore Points. The atom-type assignment in this descriptor is the hydrogen-bond donor.

SpMax2_Bh(m), SpMax3_Bh(m), *largest eigenvalue n. 2 and n.3 of Burden matrix weighted by mass.*

CIC0, *Complementary Information Content index (neighborhood symmetry of 0-order):* CIC0 is an information index and measures the deviation of the information content IC*k* from its maximum value.

$$\mathrm{CIC}k = \log_2 \mathrm{nAT} - \mathrm{IC}k$$

Figure 27. Equation to obtain CIC0 (Source: KODE Chemoinformatics)

Being nAT the total number of molecule atoms.

SM1_Dz(Z), *spectral moment of order 1 from Barysz matrix weighted by atomic number:* This value is calculated from

$$\mathrm{SM1\_M}(\mathrm{w}) = \mathrm{sgn}\left(\sum_{i=1}^{\mathrm{nSK}} \lambda_i\right) \cdot \ln\left(1 + \left|\sum_{i=1}^{\mathrm{nSK}} \lambda_i\right|\right)$$

Figure 28. Equation to obtain SM1_Dz(m). (Source: KODE Chemoinformatics)

In this case the formula is applied with Barysz matrices.

GATS1i, *Geary autocorrelation of lag 1 weighted by ionization potential:* GATS1i is a 2D autocorrelation, a descriptor which describes how a considered property is distributed along a topological molecular structure. In GATS case is calculated by:

$$\mathrm{GATS}kw = \frac{\frac{1}{2\Delta} \cdot \sum_{i=1}^{\mathrm{nAT}} \sum_{j=1}^{\mathrm{nAT}} \delta_{ij} \cdot \left(w_i - w_j\right)^2}{\frac{1}{(\mathrm{nAT}-1)} \cdot \sum_{i=1}^{\mathrm{nAT}} \left(w_i - \bar{w}\right)^2}$$

Figure 29.Equation to obtain GATS1i. (Source: KODE Chemoinformatics)

Being the property calculated, in this case, the carbon-scaled atomic ionization potential.

NdsCH, *Number of atoms of type dsCH:* This new attribute belongs to the Atom-type E-state indices, which are molecular descriptors that combine structural information about the electron accessibility associated with each atom-type, an indication of the presence or absence of a given atom-type and a count of the number of atoms of a given atom-type. In this case is a counting of the atom-type =CH-.

NdssC, *Number of atoms of type dssC:* This attribute is a counting of the atom type =C<.

nC(=N)N2, *number of guanidine derivatives*. This attribute is the sum of all the sp2-hybridized C belonging to a conjugated system, in this case (=N)N2.

SpPosA_B(v), *normalized spectral positive sum from Burden matrix weighted by van der Waals volume.*

$$\mathrm{SpPosA\_M}(\mathrm{w}) = \frac{1}{\mathrm{nSK}} \cdot \sum_{i=1}^{n^+} \left(\lambda_i^+\right)$$

Figure 30. Equation to obtain SpPosA_B(v)i. (Source: KODE Chemoinformatics)

Nroh, *number of hydroxyl groups*.

nOHp, *number of primary alcohols.*

O-056, This attribute is a simple molecular descriptor defined as the number of specific atom types in a molecule. They are calculated on the basis of molecular composition and atom connectivity. In this case the specific atom type is the alcohol.

B03[O-O], *Presence/absence of O - O at topological distance 3.*

F03[O-O], *Frequency of O - O at topological distance 3*

GATS2i. This molecular descriptor belongs to the 2D Autocorrelation group. In this specific case, it is a Geary autocorrelation measuring the carbon-scaled atomic ionization potential.

$$\mathrm{GATS}kw = \frac{\frac{1}{2\Delta} \cdot \sum_{i=1}^{\mathrm{nAT}} \sum_{j=1}^{\mathrm{nAT}} \delta_{ij} \cdot \left(w_i - w_j\right)^2}{\frac{1}{(\mathrm{nAT}-1)} \cdot \sum_{i=1}^{\mathrm{nAT}} \left(w_i - \bar{w}\right)^2}$$

Figure 31. Geary autocorrelation

SssS. Sum of the electro topological state of all S atoms.

ATS4m. It is also a 2D autocorrelation feature which in this case the property measured is the carbon-scaled atomic mass through the Centred Broto-Moreau autocorrelation.

$$ATS_{kw} = \sum_{i=1}^{nAT-1} \sum_{j=i+1}^{nAT} w_i \cdot w_j \cdot \delta_{ij}$$

Figure 32. Broto-Moreau autocorrelation

EE_B(m). This feature is a 2D matrix-based descriptor. In this case, the matrix that is used is the Burden matrix and the algebraic operator is the Estrada-like index, which formula can be seen below.

$$\mathrm{EE\_M(w)} = \ln\left(1 + \sum_{i=1}^{\mathrm{nSK}} e^{\lambda_i}\right)$$

Figure 33. Equation to obtain EE_b(m)

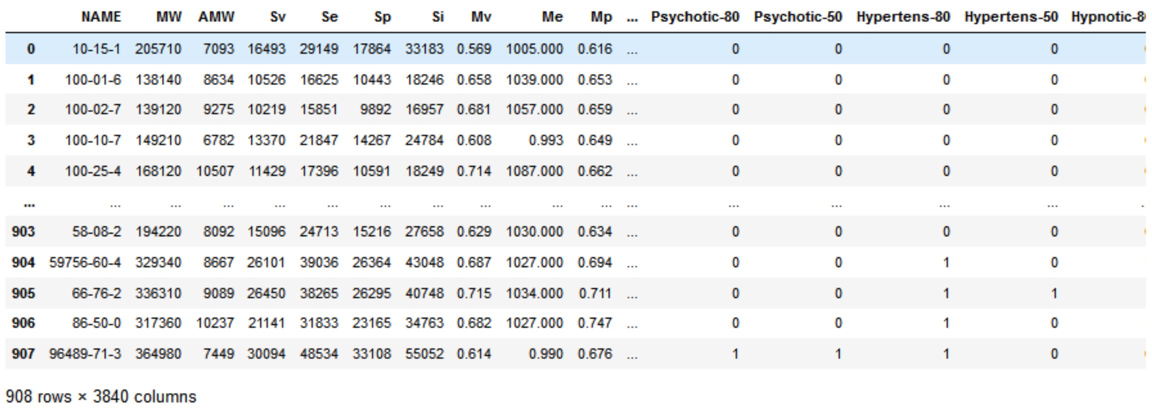
2. Methodology

2.1. Data set pre-treatment

A dataset with 3839 molecular descriptors and 908 samples needs a first prefiltering to eliminate the existing constant values for all the chemicals and the missing values. If this treatment it is not done, the training time and the performance of the algorithms created can be highly affected because they dependence on the features. We should retain the features that can really help our machine learning models.

With datasets of this type, they are several advantages if this pre-treatment is done:

- Data redundancy is reduced, reducing the training time.
- Final models have a higher explainability.
- Overfitting reduced.



	NAME	MW	AMW	Sv	Se	Sp	Si	Mv	Me	Mp	...	Psychotic-80	Psychotic-50	Hypertens-80	Hypertens-50	Hypnotic-8
0	10-15-1	205710	7093	16493	29149	17864	33183	0.569	1005.000	0.616	...	0	0	0	0	
1	100-01-6	138140	8634	10526	16625	10443	18246	0.658	1039.000	0.653	...	0	0	0	0	
2	100-02-7	139120	9275	10219	15851	9892	16957	0.681	1057.000	0.659	...	0	0	0	0	
3	100-10-7	149210	6782	13370	21847	14267	24784	0.608	0.993	0.649	...	0	0	0	0	
4	100-25-4	168120	10507	11429	17396	10591	18249	0.714	1087.000	0.662	...	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
903	58-08-2	194220	8092	15096	24713	15216	27658	0.629	1030.000	0.634	...	0	0	0	0	
904	59756-60-4	329340	8667	26101	39036	26364	43048	0.687	1027.000	0.694	...	0	0	1	0	
905	66-76-2	336310	9089	26450	38265	26295	40748	0.715	1034.000	0.711	...	0	0	1	1	
906	86-50-0	317360	10237	21141	31833	23165	34763	0.682	1027.000	0.747	...	0	0	1	0	
907	96489-71-3	364980	7449	30094	48534	33108	55052	0.614	0.990	0.676	...	1	1	1	0	

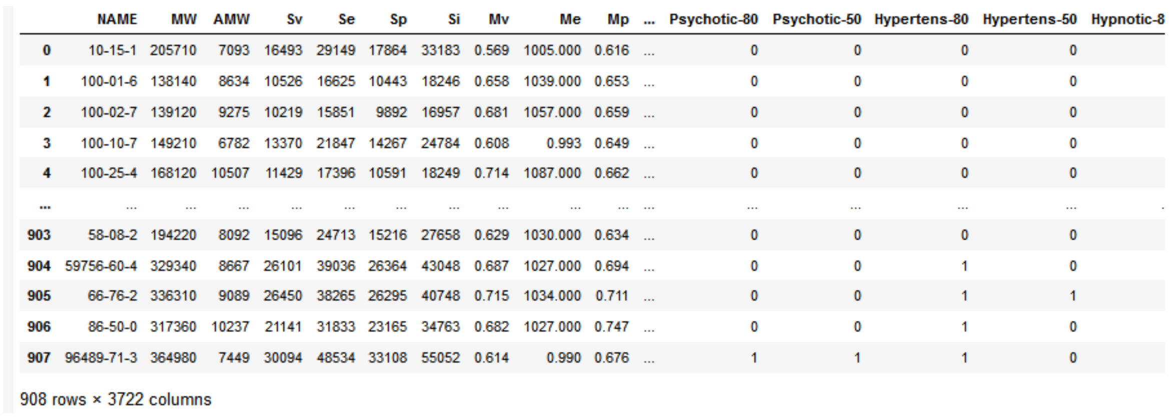
908 rows × 3840 columns

Figure 34. Dataset without toxicity

First, a removing of the columns with missing values is done through the code:

datanona=data.dropna(axis='columns')

Obtaining a reduction to 3721 attributes.

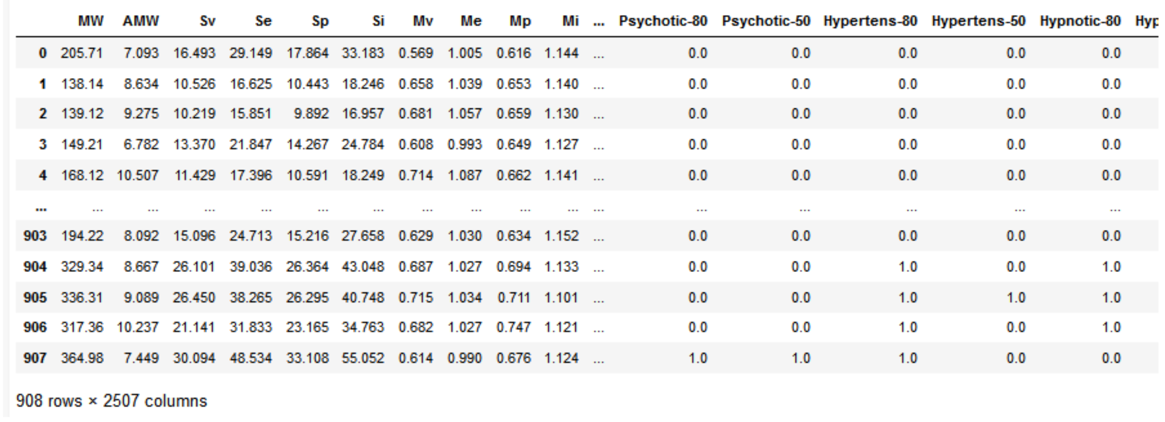


	NAME	MW	AMW	Sv	Se	Sp	Si	Mv	Me	Mp	...	Psychotic-80	Psychotic-50	Hypertens-80	Hypertens-50	Hypnotic-8
0	10-15-1	205710	7093	16493	29149	17864	33183	0.569	1005.000	0.616	...	0	0	0	0	
1	100-01-6	138140	8634	10526	16625	10443	18246	0.658	1039.000	0.653	...	0	0	0	0	
2	100-02-7	139120	9275	10219	15851	9892	16957	0.681	1057.000	0.659	...	0	0	0	0	
3	100-10-7	149210	6782	13370	21847	14267	24784	0.608	0.993	0.649	...	0	0	0	0	
4	100-25-4	168120	10507	11429	17396	10591	18249	0.714	1087.000	0.662	...	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
903	58-08-2	194220	8092	15096	24713	15216	27658	0.629	1030.000	0.634	...	0	0	0	0	
904	59756-60-4	329340	8667	26101	39036	26364	43048	0.687	1027.000	0.694	...	0	0	1	0	
905	66-76-2	336310	9089	26450	38265	26295	40748	0.715	1034.000	0.711	...	0	0	1	1	
906	86-50-0	317360	10237	21141	31833	23165	34763	0.682	1027.000	0.747	...	0	0	1	0	
907	96489-71-3	364980	7449	30094	48534	33108	55052	0.614	0.990	0.676	...	1	1	1	0	

908 rows × 3722 columns

Figure 35. Data set after removing columns with missing values

The next step is to remove the constant features, that are a lot and they don't provide any type of information that can help the classification and regression. After this is done, the number of attributes left is 2507.



	MW	AMW	Sv	Se	Sp	Si	Mv	Me	Mp	Mi	...	Psychotic-80	Psychotic-50	Hypertens-80	Hypertens-50	Hypnotic-80	Hyp
0	205.71	7.093	16.493	29.149	17.864	33.183	0.569	1.005	0.616	1.144	...	0.0	0.0	0.0	0.0	0.0	
1	138.14	8.634	10.526	16.625	10.443	18.246	0.658	1.039	0.653	1.140	...	0.0	0.0	0.0	0.0	0.0	
2	139.12	9.275	10.219	15.851	9.892	16.957	0.681	1.057	0.659	1.130	...	0.0	0.0	0.0	0.0	0.0	
3	149.21	6.782	13.370	21.847	14.267	24.784	0.608	0.993	0.649	1.127	...	0.0	0.0	0.0	0.0	0.0	
4	168.12	10.507	11.429	17.396	10.591	18.249	0.714	1.087	0.662	1.141	...	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
903	194.22	8.092	15.096	24.713	15.216	27.658	0.629	1.030	0.634	1.152	...	0.0	0.0	0.0	0.0	0.0	
904	329.34	8.667	26.101	39.036	26.364	43.048	0.687	1.027	0.694	1.133	...	0.0	0.0	1.0	0.0	1.0	
905	336.31	9.089	26.450	38.265	26.295	40.748	0.715	1.034	0.711	1.101	...	0.0	0.0	1.0	1.0	1.0	
906	317.36	10.237	21.141	31.833	23.165	34.763	0.682	1.027	0.747	1.121	...	0.0	0.0	1.0	0.0	1.0	
907	364.98	7.449	30.094	48.534	33.108	55.052	0.614	0.990	0.676	1.124	...	1.0	1.0	1.0	0.0	0.0	

908 rows × 2507 columns

Figure 36. Final feature dataset

This final data is also standardized to do the regression with min max scalers from -1 to 1 and the standardization scale.

The output variable in regression problems, LC50 96h, also needs a treatment because as can be seen in the figures below, there are some values very high that can produce big errors.

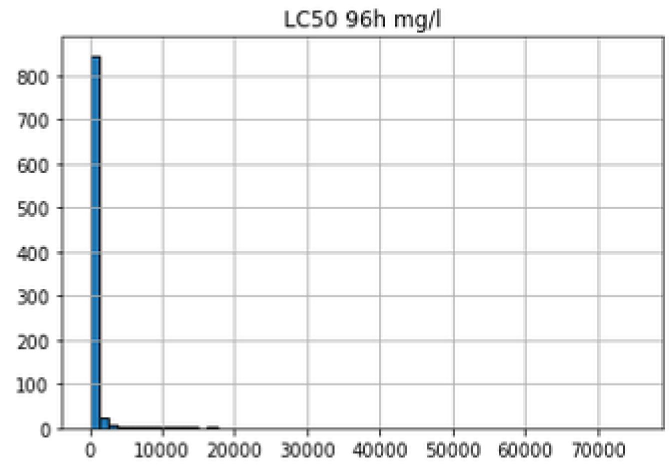


Figure 37. Histogram of the LC50 96h

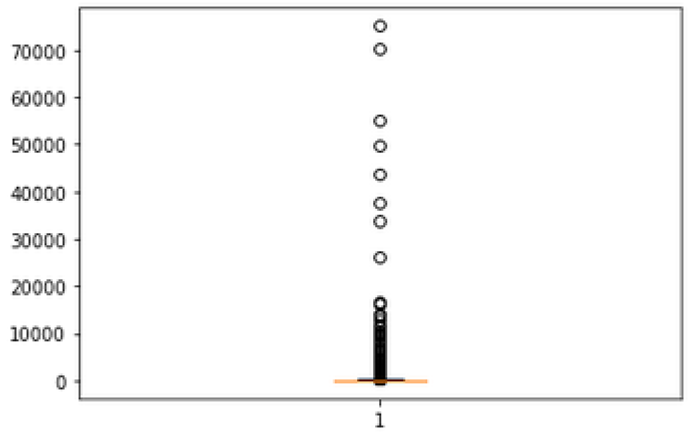


Figure 38. Boxplot of the LC50 96h

To solve this, a natural logarithm has been applied on the data obtaining a more usable data. From the figure below, it can be seen that the variable LC50 96h has a log-normal distribution.

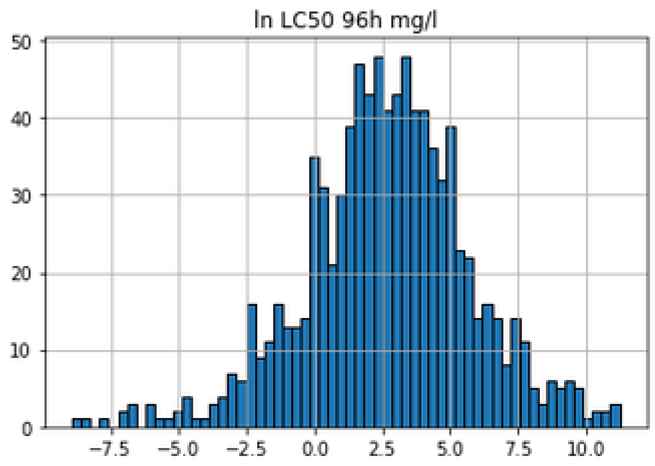


Figure 39. Histogram of ln(Y)

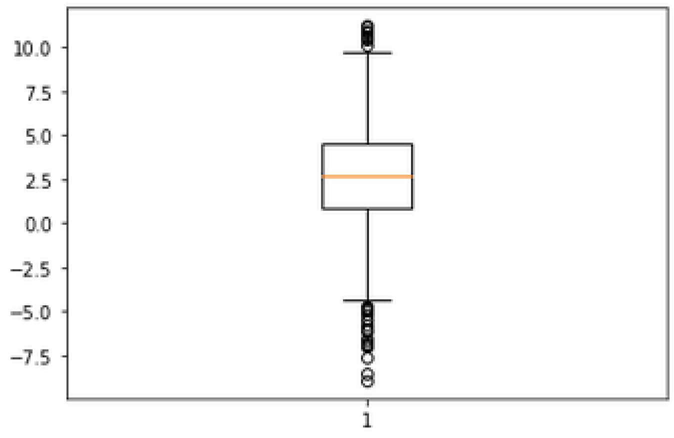


Figure 40. Boxplot of ln(Y)

The output variable for the classification is divided in 4 categories depending on the LC50 96h. If the LC50 96h is lower than 1mg/l, then, the chemical toxicity is categorized as Acute 1, if the value is between 1 mg/l and 10 mg/l is Acute 2, if it is between 10 mg/l and 100 mg/l is Acute 3 and finally it the value is above 100 mg/l, the category is Not classified.

Knowing this, the order from the most toxic category to the least toxic category is Acute 1, Acute 2, Acute 3 and No Classified.

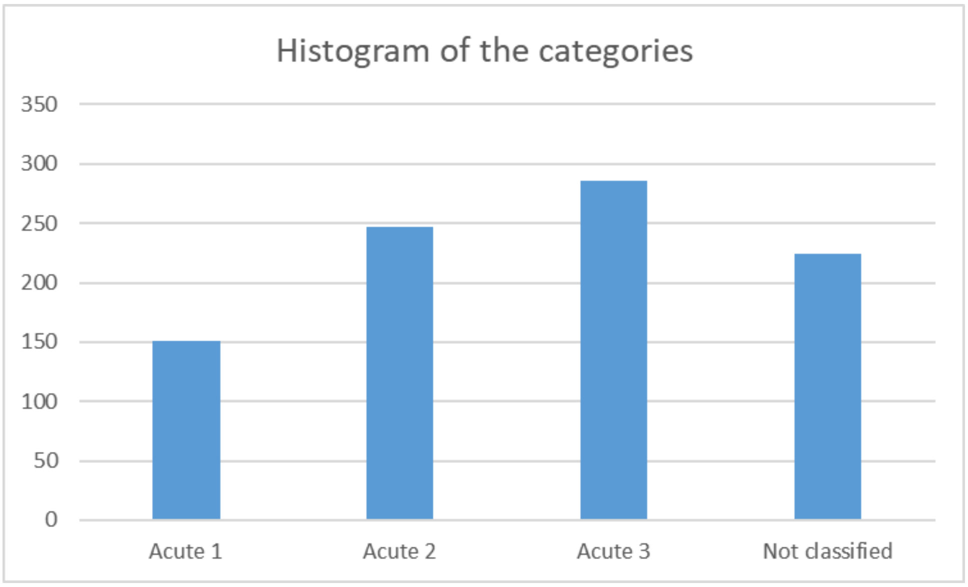


Figure 41. Histogram of the categories

2.2. Feature selection

Once the final dataset is ready it is important to do a selection of the best features for classification and regression problems. This process automatically selects the features contained in the data set that can contribute most to the final prediction.

This selection can reduce the overfitting, this means less chances to let the noise contribute to the prediction, it also can improve the final accuracy and having less features to process means a reduction of the training time

2.2.1. Feature selection for classification

The first feature selection process for classification is done by ANOVA (Analysis of Variance), which is a statistical method commonly used to check the means of groups different from each other. This method uses the F-test to check for these mean differences. The code to implement ANOVA on python is: [16]

```
from sklearn.feature_selection import SelectKBest

from sklearn.feature_selection import f_classif

fs = SelectKBest(score_func=f_classif, k=8)

X_selected = fs.fit_transform(X, Y)
```

Being k the number of attributes wanted, in this case 8, and f_classif the computation of ANOVA.

Another method used is the Mutual information, which is already used and explained for regression.

The code to implement it is:

```
from sklearn.feature_selection import mutual_info_classif

sel_mutual = SelectKBest(mutual_info_classif, k=8)

X_train_mutual = sel_mutual.fit_transform(X, Y)
```

Being k the number of features wanted.

The third method is the Sequential feature selector, in this case, in forward selection. SFS adds the number of features predefined to form a feature subset. There exists the backward selection that eliminates features from the initial set but it is impossible to use it in this project because to eliminate 2500 features would take weeks to complete.

To implement this method on python:

```
from mlxtend.feature_selection import SequentialFeatureSelector

from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_jobs=-1)

feature_selector = SequentialFeatureSelector(clf,

        k_features=5,

        forward=True,

        floating=False,

        verbose=2,
    scoring='accuracy',

    cv=5)
```

In this case, the criterion function is Random Forest Classifier and the number of features wanted are 5. The cv=5 represents the cross-validation splitting strategy (5 folds in this case)

The last feature selection method is the Recursive Feature Elimination (RFE). This algorithm is easy to configure, is good at selecting the feature that have more impact predicting the target variable and it is easy to use, this is why this algorithm is one of the most popular. This algorithm needs the number of features to select and the algorithm choose, in this case, random tree classifier.

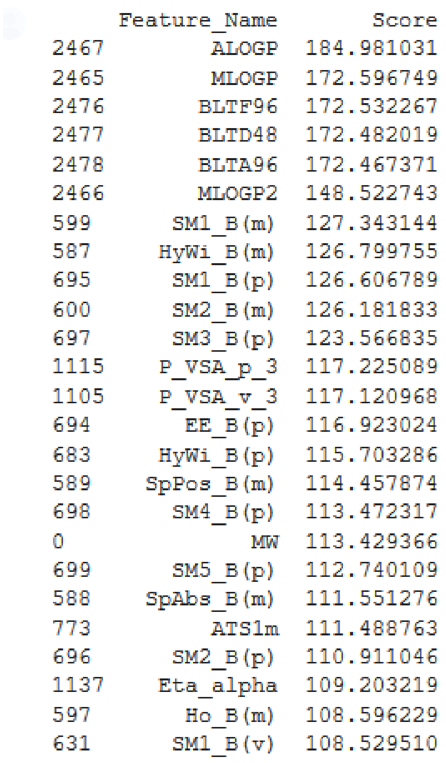
This method works first training the estimator on an initial set of features, then, the features with lower impact on the output variable are pruned. This procedure is repeated until in the set remind the number of features selected.

The code for its implementation is:

```
from sklearn.feature_selection import RFE

clf3 = RandomForestClassifier(n_jobs=-1)

rfe = RFE(clf3, n_features_to_select=5)

Xrfe=rfe.fit(X,Y)
```

2.2.1.1. Features obtained for classification

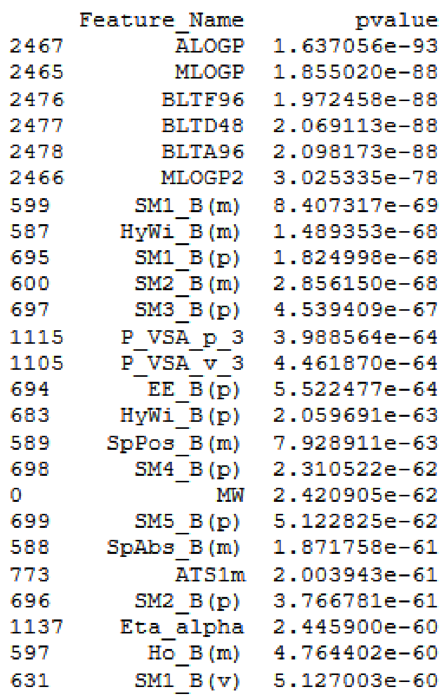
Once the univariate method (ANOVA) for all the features is applied, the scores of the 25 best features are:



```
        Feature_Name       Score
2467           ALOGP  184.981031
2465           MLOGP  172.596749
2476          BLTF96  172.532267
2477          BLTD48  172.482019
2478          BLTA96  172.467371
2466          MLOGP2  148.522743
599          SM1_B(m)  127.343144
587         HyWi_B(m)  126.799755
695          SM1_B(p)  126.606789
600          SM2_B(m)  126.181833
697          SM3_B(p)  123.566835
1115        P_VSA_p_3  117.225089
1105        P_VSA_v_3  117.120968
694           EE_B(p)  116.923024
683         HyWi_B(p)  115.703286
589        SpPos_B(m)  114.457874
698          SM4_B(p)  113.472317
0                  MW  113.429366
699          SM5_B(p)  112.740109
588        SpAbs_B(m)  111.551276
773            ATS1m  111.488763
696          SM2_B(p)  110.911046
1137        Eta_alpha  109.203219
597           Ho_B(m)  108.596229
631          SM1_B(v)  108.529510
```

Figure 42. Scores from the 25 better features.

And the smallest p_values are:



```
        Feature_Name        pvalue
2467           ALOGP  1.637056e-93
2465           MLOGP  1.855020e-88
2476          BLTF96  1.972458e-88
2477          BLTD48  2.069113e-88
2478          BLTA96  2.098173e-88
2466          MLOGP2  3.025335e-78
599          SM1_B(m)  8.407317e-69
587         HyWi_B(m)  1.489353e-68
695          SM1_B(p)  1.824998e-68
600          SM2_B(m)  2.856150e-68
697          SM3_B(p)  4.539409e-67
1115        P_VSA_p_3  3.988564e-64
1105        P_VSA_v_3  4.461870e-64
694           EE_B(p)  5.522477e-64
683         HyWi_B(p)  2.059691e-63
589        SpPos_B(m)  7.928911e-63
698          SM4_B(p)  2.310522e-62
0                  MW  2.420905e-62
699          SM5_B(p)  5.122825e-62
588        SpAbs_B(m)  1.871758e-61
773            ATS1m  2.003943e-61
696          SM2_B(p)  3.766781e-61
1137        Eta_alpha  2.445900e-60
597           Ho_B(m)  4.764402e-60
631          SM1_B(v)  5.127003e-60
```

Figure 43. Pvalues from the 25 better features.

After knowing these scores and pvalues, it has been decided to choose a number k (number of features to choose) between 3 to 8 in all methods of feature selection.

For classification, 9 groups of features have been obtained from the methods described before, the ANOVA method, the Mutual Information method and the RFE method (applying changes in the number of features desired k):

Table 2. Features obtained from classification feature selection

Features 1	HyWi_B(m)
	SM1_B(m)
	MLOGP
	MLOGP2
	ALOGP
	BLTF96
	BLTD48
	BLTA96
Features 2	SM1_B(m)
	MLOGP
	MLOGP2
	ALOGP
	ALOGP2
	BLTF96
	BLTD48
	BLTA96
Features 3	H_Dz(m)
	P_VSA_ppp_D
	nC(=N)N2
	ALOGP
	BLTA96
Features 4	SpMax2_Bh(m)
	SpMax3_Bh(m)
	MLOGP
	ALOGP
	ALOGP2
Features 5	CIC0
	SM1_Dz(Z)
	GATS1i
	NdsCH
	NdssC
	MLOGP

Features 6	MLOGP
	ALOGP
	BLTF96
	BLTD48
	BLTA96
Features 7	MLOGP2
	ALOGP
	ALOGP2
	BLTD48
	BLTA96
Features 8	MLOGP
	ALOGP
	BLTF96
Features 9	ALOGP
	ALOGP2
	BLTA96

2.2.2. Feature selection for regression

To do the feature selection for the regression problems, two methods have been carried out. The first one is using the Pearson's Correlation Coefficient via the f_regression function which is a linear model with the objective of testing the individual effect of many regressors [17]. The code in python is the following:

```
from sklearn.feature_selection import SelectKBest

from sklearn.feature_selection import f_regression

fs = SelectKBest(score_func=f_regression, k=6)

X_selected = fs.fit_transform(X, Y)
```

Being k the number of features desired.

The last method used is the mutual informal information.This one consists on the calculation of the mutual information value for each feature with respect to the output variable selecting the features that have a higher information gain. This means that the dependency of the features with the target value is measured. To implement it in python:

```
from sklearn.feature_selection import mutual_info_regression

f_selector = SelectKBest(score_func=mutual_info_regression, k=6)

f_selector.fit(X, Y)

X_fs = f_selector.transform(X)
```

Being k, the number of features wanted.

2.2.2.1. Features obtained for regression

After the application of the methods above mentioned, different groups of features have been obtained and trained. All this groups can be seen in the table below.

Table 3 Features obtained from feature selection for regression

Features 1	SpPosA_B(v)
	nROH
	nOHp
	O-056
	B03[O-O]
	F03[O-O]
	MLOGP
	BLTF96
Features 2	SM1_B(m)
	MLOGP
	MLOGP2
	ALOGP
	ALOGP2
	BLTF96
	BLTD48
	BLTA96
Features 3	CIC0
	SM1_Dz(Z)
	GATS1i
	NdsCH
	NdssC
	MLOGP

Features 4	MLOGP
	MLOGP2
	ALOGP
	BLTF96
	BLTD48
	BLTA96
Features 5	MLOGP
	ALOGP
	ALOGP2
	BLTF96
	BLTD48
	BLTA96
Features 6	MLOGP
	ALOGP
	BLTD48
Features 7	MLOGP
	ALOGP
	ALOGP2
	BLTA96

2.3. Classification

For the classify some of the models described above are applied. In this section the code used for each method in a general format, then, for each method, 9 combinations of features are trained. For all the classification models, a split of the dataset has been made, obtaining the training set (80% of total rows) and the test set (20%)

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size= 0.2, random_state=0)
```

The random state parameter indicates that the split is not random so, for all the models, the same split is made.

2.3.1. K-NN

For the k-NN, the code to obtain the accuracy of the training and the testing set:

```
n_neighbors = 15

knn = KNeighborsClassifier(n_neighbors)

knn.fit(X_train, Y_train)
```

```
print('Accuracy of K-NN classifier on training set: {:.2f}'
    .format(knn.score(X_train, Y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'
    .format(knn.score(X_test, Y_test)))
```

From an adaptation of the code above, it is possible to asses all k to determine which one will have higher accuracy from a plot similar to the figure above.

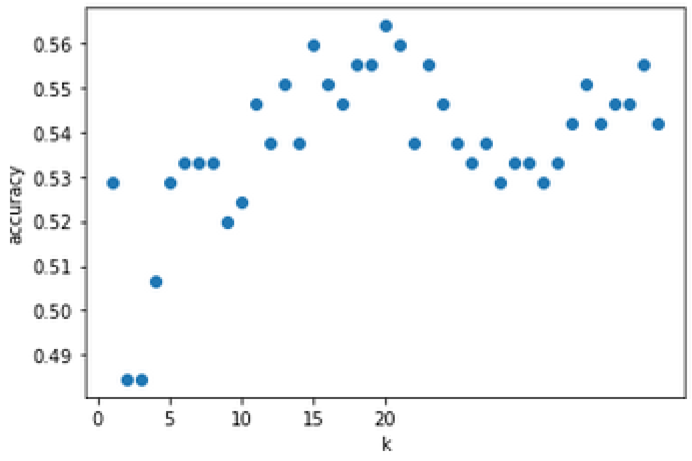


Figure 44. assessment of the accuracy of all k

It is also possible to compare the evolution of the train accuracy and the test accuracy when the number of k is increased.

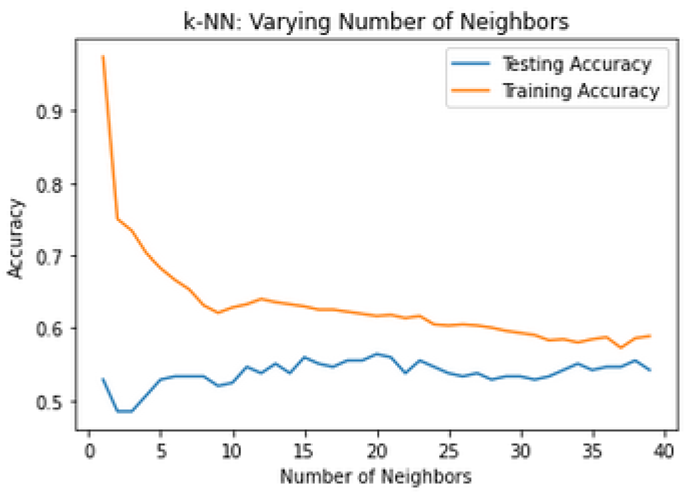


Figure 45. Accuracy varying the number of neighbors

And to obtain the confusion matrix of the k-NN application:

pred = knn.predict(X_test)

print(confusion_matrix(Y_test, pred))

print(classification_report(Y_test, pred))

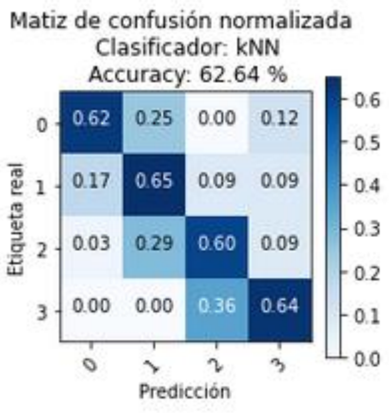


Figure 46. Confusion matrix example

2.3.2. Decision tree classifier

For the basic decision tree classifier, the implementation like in k-NN is very easy:

clf = DecisionTreeClassifier()

clfpred = clf.fit(X_train,Y_train)

y_pred = clfpred.predict(X_test)

ytrain_pred = clfpred.predict(X_train)

print("Accuracy:",metrics.accuracy_score(Y_test, y_pred))

print("Accuracy:",metrics.accuracy_score(Y_train, ytrain_pred))

With this code, the accuracy for the training set and the testing set can be determined.

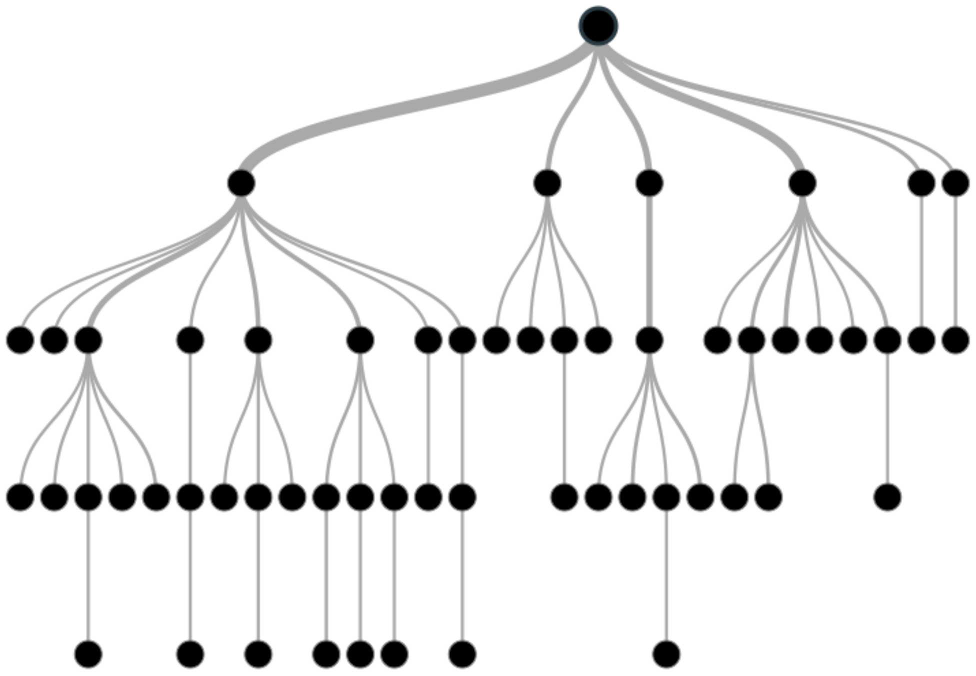


Figure 47. Decision Tree. (Source: BigML)

To improve this model, some hyperparameters can be set. To set them GridsearchCV is used, first of all, setting the parameters that are needed to be tested and then, the number of folds. In this project the hyperparameters tested of the decision tree classifier are the Criterion, the Max depth, the max leaf nodes and finally the min samples split. The implementation of GridSearchCV, in this example is:

```
from sklearn.model_selection import GridSearchCV

param_dict = {

   "criterion":['gini','entropy'],

   "max_depth":range(1,15),

   'max_leaf_nodes': list(range(2, 100)),

   'min_samples_split': range(2,20),

}

grid=GridSearchCV(DecisionTreeClassifier(),

            param_grid=param_dict,
```

cv=20,

verbose=1,

n_jobs=-1)

GSCV=grid.fit(X,Y)

The criterion parameter is the function that measures the quality of a split, for decision tree classifier it can be used the gini function or the entropy function. [18]

The max depth parameter represents the maximum depth of the tree. If this value can be omitted letting the nodes of the tree to expand until all leaves are pure.

The max leaf nodes parameter is the maximum of nodes that the tree can have.

The min samples split parameter is the minimum number of samples required to split an internal node.

2.3.3. AdaBoost

AdaBoost is more complex to implement than a basic decision tree classifier, first, is needed to define the base estimator, in this case, the decision tree classifier. Also, some hyperparameters can be tuned such as the number of estimators and the learning rate. The code used in this project is:

```
clf = DecisionTreeClassifier()

from sklearn.ensemble import AdaBoostClassifier

ada = AdaBoostClassifier(base_estimator=clf, n_estimators=100,

                learning_rate=1.5, random_state=0)

ada = ada.fit(X_train, Y_train)
```

The parameter n estimators define the number of estimators at which the boost will be ended (in case of perfect fit, the learning process will stop before).

The parameter learning rate is the weight that is applied to each classifier at each boosting iteration.

2.3.4. Gradient Boosting

In this project, the gradient boost classifier works very similar to AdaBoost in terms of hyperparameters, needing a number of estimators, the learning rate and the max depth. The code is:

```
from sklearn.ensemble import GradientBoostingClassifier

gbrt = GradientBoostingClassifier(random_state=0, n_estimators=5500,

                    max_depth=1, learning_rate=0.01)

gbrt.fit(X_train, Y_train)

print('Precisión Gradient Boosting train/test  {0:.3f}/{1:.3f}'

  .format(gbrt.score(X_train, Y_train), gbrt.score(X_test, Y_test)))
```

2.3.5. Bagging classifier

For the bagging classifier it is important to set the number of splits and repetitions desired for the cv, then the type of error score and the scoring method. The code is the following:

```
model=BaggingClassifier()

cv=RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)

n_scores = cross_val_score(model, X, Y, scoring='accuracy', cv=cv, n_jobs=-1, error_score='raise')

print('Accuracy:%.3f (%.3f)' % (mean(n_scores), std(n_scores)))
```

The Repeated Stratified K fold is a cross validator used in this project for the bagging classifier. Its parameters are:

n_splits: Represents the number of folds

n_repeats: Number of times that the cross-validator needs to be repeated.

For the evaluation of the score it has been used the function cross_val_score.

2.3.6. Random forest classifier

The random forest classifier in this project works in the same way that the Bagging classifier, it only changes the model used. It has to be said that it can be also implemented like the Decision tree classifier.

```
from sklearn.ensemble import RandomForestClassifier

model2 = RandomForestClassifier()

cv2=RepeatedStratifiedKFold(n_splits=10, n_repeats = 3, random_state=1)

n_scores2 = cross_val_score(model2, X, Y, scoring='accuracy', cv=cv2, n_jobs=-1, error_score='raise')

print('Accuracy:%.3f (%.3f)' % (mean(n_scores2), std(n_scores2)))
```

2.3.7. Gaussian Naïve Bayes

The last classification method used is the Gaussian Naïve Bayes which implementation is very easy, like k-NN. The code to follow to determine the accuracy of the training and the testing sets is:

```
gnb = GaussianNB()

gnb.fit(X_train, Y_train)

y_pred = gnb.predict(X_test)



print('Precisión en el set de Entrenamiento: {:.2f}'

    .format(gnb.score(X_train, Y_train)))

print('Precisión en el set de Test: {:.2f}'

    .format(gnb.score(X_test, Y_test)))
```

2.4. Regression

For the regression, 7 combinations of attributes are tested using different methods of standardization, min max and standard scale. To assess the predictability of the models obtained, the mean absolute error is used, which formula can be seen in the figure below.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Figure 48. MAE equation

With MAE it is difficult to see how accurate are the models so, it is needed a normalization.

$$NMAE = \frac{MAE}{Ymax - Ymin}$$

For the regression models, as for classification, the set has been split in a training set and a test set.

2.4.1. Decision tree regressor

The decision tree regressor application in Python is similar to the decision tree classifier seen before sharing the majority of hyperparameters. The change is the criterion parameter which is the function that measures the quality of a split and tries to minimize the L2 error if mse criterion is used or, like in this case, the L1 error, using mae criterion. The basic code can be seen below:

```
DecisionTreeRegModel=DecisionTreeRegressor(criterion = "mae")

DecisionTreeRegModel.fit(X_train,Y_train)

y_pred = DecisionTreeRegModel.predict(X_test)
```

2.4.2. Bagginig regressor

Like the Decision tree regressor, the Bagging regressor shares many similarities with the Bagging classifier when it comes to implementing it in the program python being the basic change the model used.

```
model = BaggingRegressor()

cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)

n_scores = cross_val_score(model, X, Y, scoring='neg_mean_absolute_error', cv=cv, n_jobs=-1, error_score='raise')

print('MAE: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))
```

2.4.3. Gradient boosting regressor

The code to implement the gradient boosting regressor is:

```
params = {'n_estimators': 500,
        'max_depth': 4,
        'min_samples_split': 5,
        'learning_rate': 0.01,
        'loss': 'ls'}
reg = ensemble.GradientBoostingRegressor(**params)
reg.fit(X_train, Y_train)
```

2.4.4. Random Forest Regressor

The random forest regressor is implemented in the same way that the decision tree regressor:

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators = 1000, random_state = 0)# Train the model on training data
rf.fit(X_train, Y_train);
Y_predictions = rf.predict(X_test)# Calculate the absolute errors
```

The number of estimators determines whit how many decision trees are initialised the random forest.

2.4.5. k-NN regressor

k-NN regressor has a model to implement it directly in Python, it is only necessary to specify the hyperparameters, in this case the number of neighbours.

```
from sklearn.neighbors import KNeighborsRegressor
neigh = KNeighborsRegressor(n_neighbors=17)
neigh.fit(X_train, Y_train)
```

2.4.6. Support vector regressor

The code for the implementation of a basic support vector regressor is the following:

```
from sklearn.svm import SVR
```

regressor = SVR()

regressor.fit(X_train, Y_train)

2.4.7. Artificial Neural Network

The artificial Neural Network in this project has 2 hidden layers of 32 nodes each one and the code is the following for a regression problem (loss and metrics are calculated with MAE and MSE.

model = Sequential()

model.add(Dense(32, input_dim=8, kernel_initializer='normal', activation='relu'))

model.add(Dense(32, activation='relu'))

model.add(Dense(1, activation='linear'))

model.compile(loss='mse', optimizer='adam', metrics=['mse','mae'])

history=model.fit(X_train, Y_train, epochs=50, batch_size=10, verbose=1,validation_split=0.2)

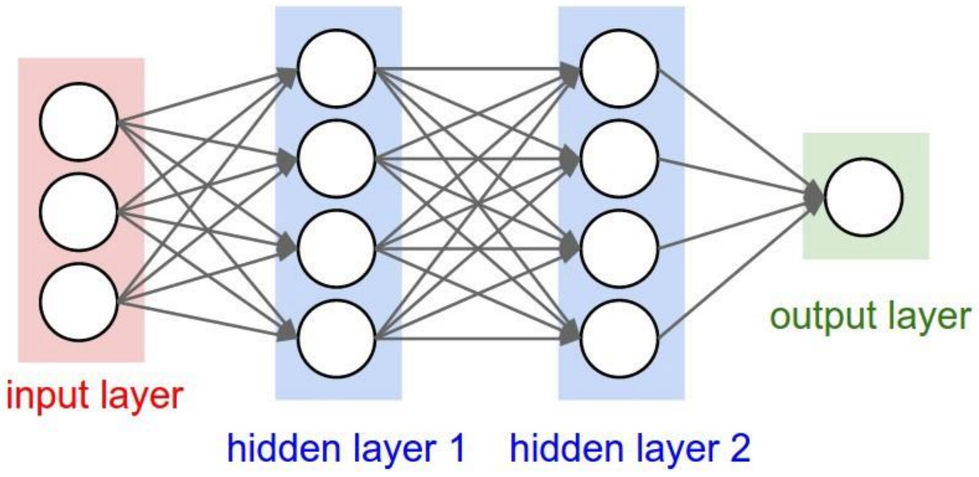


Figure 49. ANN with 2 hidden layers. (Source: Coding Neural Network)

Dense is the most common and frequently used layer in artificial neural networks and its parameters are:

Input_dim: represents the number of dimensions of the input data.

Kernel initializer: is the parameter that to initialise statistically (in this case with normal function) the weights in the model.

Activation: in this parameter is set the function that will define the output of a node given an input or a set of inputs. Relu is the most used activation function since it is used in almost all neural networks.

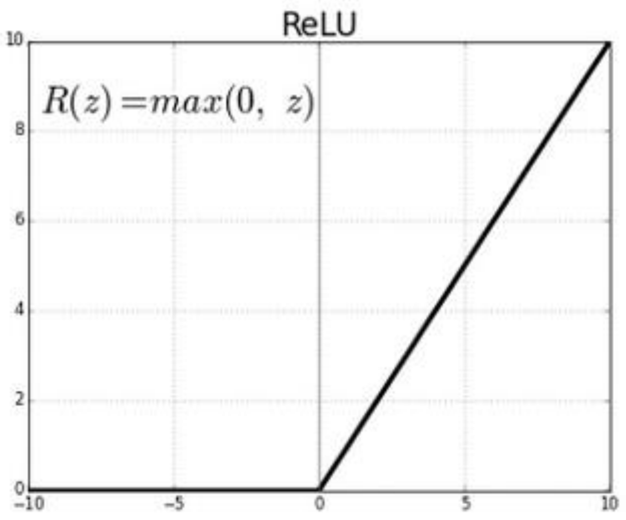


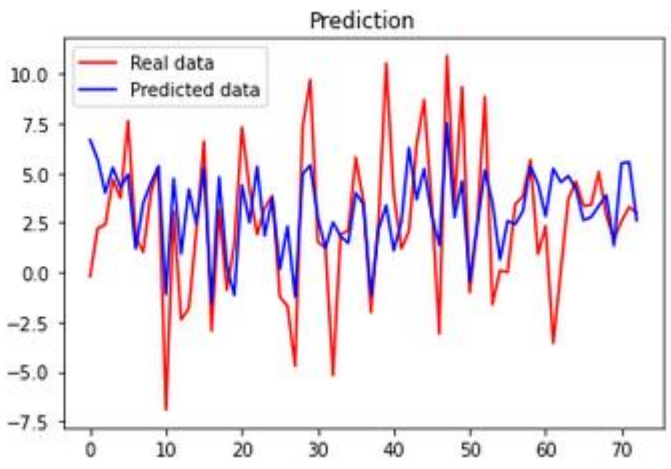
Figure 50. Relu activation function. (Source: Towardsscience)

For the compilation of the model, the optimizer is one of the two arguments required. In this case it is used the Adam optimizer which optimization is a stochastic gradient descent method. [19]

Loss parameter is also used in the compilation. This parameter, as it names indicates, is the function to evaluate a candidate solution, trying to minimize the error.

Metric functions are very similar to loss function (it also judges the performance of the model), the difference resides in that the results used to evaluate the metric are not used during the training of the model.

Figure 51. Plot of the accuracy of the ANN



3. RESULTS

3.1. Classification results

Once all the classification models before described are applied with all the feature groups, the accuracies obtained can be seen below. It is important to say that the results are obtained after the application of the GridSearchCV methods to obtain the best hyperparameters

Table 4. Table of test accuracies

	k-NN	Decision tree	AdaBoost	Gradient Boost	Bagging Classifier	Random forest classifier	Gaussian Naïve Bayes
Features 1	56	59,5	57,04	57,3	55,8	56	54
Features 2	55	59,6	53,3	56,8	55,7	56,3	54
Features 3	53	59,2	56,8	58,1	56,2	59,6	51
Features 4	61	59,24	53,3	55,5	55	58,8	57
Features 5	59	57,4	54,6	51,1	54,1	58,7	53
Features 6	55	58,7	54,8	55,5	54	52,9	52
Features 7	58	58,5	56,6	55,5	53,2	55,1	55
Features 8	56	58,7	55,2	55,5	53,8	53,7	52
Features 9	56	58	55,7	55,5	53	53,9	52

From Table 4 it can be seen clearly that the classification results are very poor, only the group of features 4 is above 60% of accuracy. To see if these results can be improved, a min max scaler has been applied.

Table 5. Table of test accuracies post data standardization

	k-NN	Decision tree	AdaBoost	Gradient Boost	Bagging Classifie	dom forest classi	Gaussian Naive Bayes
Features 1	57,69	59,5	57,69	60,4	54,5	56,1	54,4
Features 2	58,79	59,6	56,04	61	55,5	56,2	58,79
Features 3	63,74	59,2	57,14	58,8	57,3	59,3	48,9
Features 4	66	59,2	58,24	61,5	55,2	60,4	43,69
Features 5	59,89	57,4	59,89	59,3	55,2	58,7	58,24
Features 6	55,49	58,5	53,85	58,8	53	55,4	50
Features 7	56,04	58,5	58,79	57,7	54,9	58,7	49,45
Features 8	57,14	58,5	55,49	58,8	53,5	55,4	56,04
Features 9	52,75	58,6	54,95	57,7	53	61	53,3

The results obtained after the data standardization are still bad but a little improve of the accuracy of the feature group 3 and 4 reaching the 66%.

The next step done to try to improve the accuracy is to see if there is so much correlation between the molecular indicators, if yes, it will be needed another pre-processing of the dataset because modelling with the "same" variables is useless.

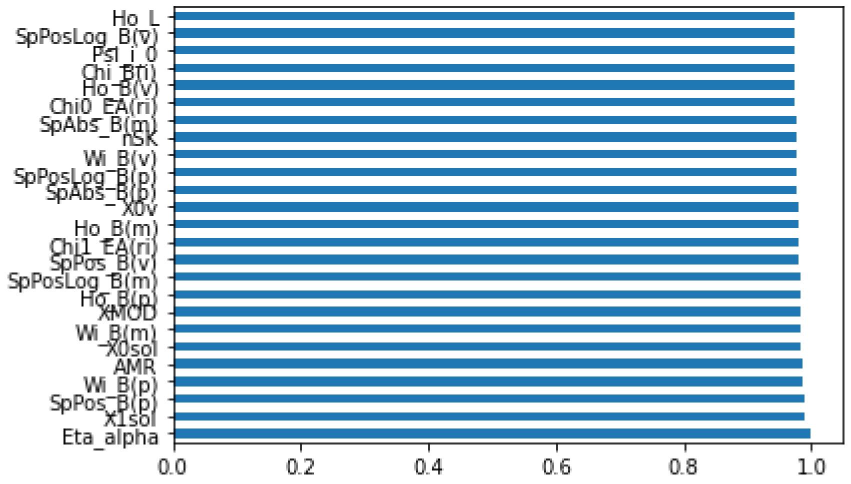


Figure 52. Correlation of Eta_alpha

After assessing the correlation of a random feature, In this case, Eta_alpha, it can be seen that there are a lot of indicators with a high correlation and there is the possibility that the dataset is a dataset highly correlated so, the next step is to clean it.

To do this, the code implemented is the following:

```
import numpy as np

corr_matrix = X.corr().abs()

upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))

to_drop = [column for column in upper.columns if any(upper[column] > 0.95)]

# Drop features

X.drop(to_drop, axis=1, inplace=True)
```

As it can be seen in the code, there are eliminated the features that have a correlation above 95%. After this, the data set is 908x1256, approximately 1000 features are eliminated. With the new set of features, the univariate method and the RFE methods are applied to select the best features obtaining the following ones:

Table 6. New group of features after pre-processing

Features 10	ALOGP
	ALOGP2
	HyWi_B(m)
	MLOGP
	MLOGP2
Features 11	HyWi_B(m)
	GATS2i
	SssS
	MLOGP
	ALOGP2

After the modelling, the results obtained can be seen below.

Table 7. Results of the new group of features

	k-NN	Decision tree	AdaBoost	Gradient Boost	Bagging Classifier	dom forest classi	Gaussian Naive Bayes
Features 10	61,54	58,9	59,34	61	55,9	57,1	59,89
Features 11	63,19	59	60,99	61,5	56,4	60,4	32,97

As it can be seen, the elimination of the features with a high correlation does not improve the final results, being them similar to the ones before obtained. A deeper pre-processing needs to be done because the high complexity of the database features.

3.2. Regression results

For the regression part, 7 groups of features with different pre-processing (min max, standard scale, and no pre-processing) are assessed. The results are given as MAE/NMAE.

Table 8. Regression results using non-pre-processed data

	Decision tree	Bagging regressor	Random forest regressor	k-NN	SVR	ANN
Features 1	2,34/0,11	1,782/0,087	1,86/0,09	2,11/0,10	2,23/0,11	3,3/0,16
Features 2	2,21/0,11	1,63/0,08	1,68/0,083	1,85/0,092	2,02/0,1	3,31/0,16
Features 3	2,06/0,10	1,52/0,076	1,89/0,093	1,61/0,08	1,81/0,089	3,33/0,16
Features 4	2,30/0,11	1,71/0,085	1,93/0,095	1,94/0,096	2,08/0,1	3,32/0,16
Features 5	2,24/0,11	1,71/0,085	1,95/0,096	1,84/0,091	2,02/0,1	3,34/0,16
Features 6	2,42/0,12	1,7/0,084	1,85/0,096	1,8/0,089	2/0,09	3,34/0,16
Features 7	2,15/0,1	1,7/0,084	1,94/0,09	1,83/0,09	2,02/0,1	3,35/0,16

Table 9. Regression results using min max standardization

	Decision tree	Bagging regressor	Random forest regressor	k-NN	SVR	ANN
Features 1	2,25/0,11	1,75/0,087	1,87/0,092	1,93/0,096	2,17/0,1	3,31/0,16
Features 2	2,13/0,1	1,67/0,08	1,74/0,086	1,96/0,097	1,87/0,093	3,2/0,15
Features 3	2,21/0,1	1,52/0,075	1,9/0,094	1,69/0,084	1,72/0,085	3,34/0,16
Features 4	2,29/0,11	1,69/0,084	1,93/0,095	2,02/0,1	2,07/0,1	3,29/0,16
Features 5	2,25/0,11	1,71/0,085	1,94/0,096	2/0,099	2,06/0,1	3,25/0,16
Features 6	2,28/0,11	1,71/0,085	1,95/0,096	2/0,099	2,02/0,1	3,23/0,16
Features 7	2,23/0,11	1,69/0,084	1,94/0,096	1,93/0,095	2,02/0,1	3,3/0,16

Table 10. Regression results using standard standardization

	Decision tree	Bagging regressor	Random forest regressor	k-NN	SVR	ANN
Features 1	2,26/0,11	1,76/0,087	1,87/0,09	1,94/0,096	2,18/0,1	3,33/0,16
Features 2	2,14/0,1	1,62/0,08	1,74/0,086	1,94/0,096	1,85/0,091	3,21/0,15
Features 3	2,26/0,11	1,51/0,076	1,9/0,094	1,71/0,085	1,73/0,086	3,57/0,17
Features 4	2,3/0,11	1,7/0,086	1,93/0,096	1,92/0,095	1,99/0,098	3,35/0,16
Features 5	2,31/0,11	1,7/0,086	1,95/0,096	1,94/0,096	1,98/0,098	3,33/0,16
Features 6	2,26/0,11	1,7/0,086	1,95/0,096	1,92/0,095	1,97/0,098	3,46/0,17
Features 7	2,34/0,11	1,69/0,086	1,95/0,096	1,88/0,093	1,96/0,097	3,33/0,16

The standard deviations can be seen at the appendix.

From the results, it can be seen that the pre-processing of this data set is not mandatory, the mean absolute errors don't vary too much. The group of features 3 is the one that obtain a higher accuracy through the Bagging regressor error, having an error (normalized mean absolute error) of 7,6%.

Like in classification, to try to improve the accuracy, the dataset with all highly correlated features is used to select new features:

Table 11. Features wihtout hard correaltion

Features 8	HyWi_B(m)
	ATS4m
	P_VSA_v_3,
	MLOGP
	MLOGP2
	ALOGP
Features 9	EE_B(m)
	SpMax3_Bh(m)
	MLOGP
	MLOGP2
	ALOGP
	ALOGP2

And their results can be seen above:

Table 12. Results of the new features

	Decision tree	Bagging regressor	Random forest regressor	k-NN	SVR	ANN
Features 8	2,4/0,11	1,62/0,073	1,82/0,082	2,16/0,098	2,13/0,097	3,59/0,16
Features 9	2,01/0,091	1,59/0,072	1,69/0,076	1,95/0,088	2,02/0,091	3,74/0,16

As it can be seen, Bagging regressor still is the best model for this data set and the elimination of the highly correlated features don’t have any impact on the final results.

On the other hand, the model that presents worst results is the Artificial Neural Network. This probably is due to the little number of samples that the dataset has (908).

Although the obtention of acceptable results, in the field of the environmental health, errors should be minimised to a much greater extent in order to replace empirical results obtained in laboratories.

Conclusions

The accuracy obtained with the classification methods establishes that no adequate models have been achieve that correctly relate the molecular descriptors of a substance to its toxicity towards the fathead minnow. To improve these results, it is necessary to simplify the database with a more powerful pre-processing by simplifying the variables and obtaining a larger number of samples.

On the other hand, the regression models do show an improvement over the classification models. The results obtained allow us to establish that although the relationship between properties such as the toxicity of a substance and the variables implemented is not known exactly, quantitative structure-activity relationships models represent a useful tool for the estimation of these parameters in the absence of empirical values from laboratory tests, without replacing them. Although an acceptable performance it is important to stay alert in cases where the model fails giving a very large difference between actual and predicted value because of the great importance of the LC50 96h value when marketing a product.

Given the need to obtain safety parameters without incurring large costs, the accuracy and the implementation of these models in the industry will depend on the development of the computational tools for the descriptors calculation and the refinement of the expressions related to the calculation of each descriptor.

Economic analysis

The economic aspect of this project is simple due to the small number of tools needed for the development of the project. In the table below, it can be seen a breakdown of the cost of the project.

Table 13. Cost of the project

Concept	Price (€)
Dragon 7 software	800
Hardware	125
Human resources	13500
Electricity	8,66
Total	14433,66

Dragon 7 is the software that allows the calculation of the different molecular descriptors for each chemical and its license for ever is 800€.

For the hardware calculation, it has been supposed 1000€ for a computer, a mouse, a keyboard and a monitor. The life of this hardware is expected to be 4 years and the duration of the project has been 6 months so, the total amount required is 125€

The salary of a student is supposed to be 30€/h for 450 hours that the whole project took.

Finally, for the electricity consumed, taking into account that on average a computer uses 2,2kWh for 8 hours, the price of a kWh is 0,07€ and a total of 450 hours, the 8,66€ is obtained.

Environmental impact

From governmental sources [20] the average emission of CO_2 is approximately 0,30kgCO_2/kWh. Taking into account that a computer requires 0,275 kWh for 450 h, the amount of CO_2 emitted is:

$$kgCO_2 = 0{,}275 * 450 * 0{,}3 = \mathbf{37{,}125\ kgCO_2}$$

Bibliography

[1] Ankley GT, Villeneuve DL. The fathead minnow in aquatic toxicology: past, present and future. Aquat Toxicol. 2006 Jun 10;78(1):91-102. doi: 10.1016/j.aquatox.2006.01.018. Epub 2006 Feb 21. PMID: 16494955. *(June 2021)*

[2]Bishop, C. M. (2006), Pattern Recognition and Machine Learning, Springer, ISBN 978-0-387-31073-2

[3] Hastie, Trevor. (2001). The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations. Tibshirani, Robert., Friedman, J. H. (Jerome H.). New York: Springer. ISBN 0-387-95284-5. OCLC 46809224

[4] Wu, Xindong; Kumar, Vipin; Ross Quinlan, J.; Ghosh, Joydeep; Yang, Qiang; Motoda, Hiroshi; McLachlan, Geoffrey J.; Ng, Angus; Liu, Bing; Yu, Philip S.; Zhou, Zhi-Hua (2008-01-01). "Top 10 algorithms in data mining". Knowledge and Information Systems. 14 (1): 1–37. doi:10.1007/s10115-007-0114-2 *(May 2021)*

[5] Kégl, Balázs (20 December 2013). "The return of AdaBoost.MH: multi-class Hamming trees" *arXiv*:*1312.6086* *(May 2021)*

[6] Joaquín Amat Rodrigo (2020), Gradient Boosting con Python. https://www.cienciadedatos.net/documentos/py09_gradient_boosting_python.html *(May 2021)*

[7] Breiman, L. Bagging predictors. *Mach Learn* **24,** 123–140 (1996). https://doi.org/10.1007/BF00058655 *(May 2021)*

[8] Breiman, L. Random Forests. *Machine Learning* **45,** 5–32 (2001). https://doi.org/10.1023/A:1010933404324 *(May 2021)*

[9] McCallum, Andrew. "Graphical Models, Lecture2: Bayesian Network Represention" https://people.cs.umass.edu/~mccallum/courses/gm2011/02-bn-rep.pdf. *(May 2021)*

[10] Matich, Damián Jorge (2001). «Redes Neuronales: Conceptos Básicos y Aplicaciones.». https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/matich-redesneuronales.pdf *(May 2021)*

[11] Tu JV. Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. J Clin Epidemiol. 1996 Nov;49(11):1225-31. doi: 10.1016/s0895-4356(96)00002-9. PMID: 8892489. (*May 2021)*

[12] Todeschini, Roberto; Consonni, Viviana (2008). Handbook of Molecular Descriptors. Wiley. ISBN 978-3-527-29913-3.

[13] Mauri A., Consonni V., Todeschini R. (2017) Molecular Descriptors. In: Leszczynski J., Kaczmarek-Kedziera A., Puzyn T., G. Papadopoulos M., Reis H., K. Shukla M. (eds) Handbook of Computational Chemistry. Springer, Cham. https://doi.org/10.1007/978-3-319-27282-5_51

[14] Moriguchi I, Hirono S, Liu Q, Nakagome Y, Matsushita Y. (1992) Simple method of calculating octanol/water partition coefficient. Chemical & Pharmaceutical Bulletin, 40, 127-130; (b) Moriguchi I, Hirono S, Nakagome I, Hirano H. (1994) Comparison of reliability of log P values for drugs calculated by several methods. Chemical & Pharmaceutical Bulletin, 42, 976-978

[15] A.K. Ghose and G.M. Crippen, J. Comput. Chem. 1986, 7, 565-577

[16] Machine Learning Mastery. *Feature Selection For Machine Learning in Python.* *https://machinelearningmastery.com/feature-selection-machine-learning-python/ (May 2021)*

[17] Scikit Learn. *sklearn.feature_selection.f_regression.* *https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_regression.html (May 2021)*

[18] Scikit Learn. *sklearn.tree.DecisionTreeClassifier.* *https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html (May 2021)*

[19] Keras. *Optimizers. https://keras.io/api/optimizers/ (May 2021)*

[20] Ministerio para la transición Ecológica y el reto demográfico. *FACTORES DE EMISIÓNDECO2 Y COEFICIENTES DE PASO A ENERGÍA PRIMARIA* https://energia.gob.es/es-es/Paginas/index.aspx. *(May 2021)*

APPENDIX

Standard deviation

Table 14. Standard deviation of the errors using non-pre-processed data

	Decision tree	Bagging regressor	Random forest regressor	k-NN	SVR	ANN
Features 1	3,04	0,15	1,78	2,74	2,9	4,14
Features 2	3,15	0,18	1,7	2,45	2,65	4,2
Features 3	3,31	0,13	1,77	2,17	2,4	4,25
Features 4	3,28	0,19	1,75	2,52	2,7	4,22
Features 5	3,14	0,18	1,77	2,37	2,66	4,26
Features 6	3,38	0,2	1,77	2,38	2,62	4,25
Features 7	3,09	0,19	1,75	2,4	2,66	4,2

Table 15.Standard deviation of the errors using min max standardization

	Decision tree	Bagging regressor	Random forest regressor	k-NN	SVR	ANN
Features 1	3,07	0,16	1,79	2,57	2,83	4,25
Features 2	3,11	0,16	1,65	2,58	2,5	4,09
Features 3	3,11	0,14	1,77	2,27	2,34	4,24
Features 4	3,23	0,18	1,75	2,61	2,71	4,17
Features 5	3,21	0,2	1,77	2,6	2,7	4,09
Features 6	3,21	0,18	1,77	2,58	2,67	4,11
Features 7	3,3	0,178	1,76	2,51	2,66	4,17

Table 16. Standard deviation of the errors using standard standardization

	Decision tree	Bagging regressor	Random forest regressor	k-NN	SVR	ANN
Features 1	3,06	0,15	1,79	2,54	2,96	4,26
Features 2	3,11	0,2	1,65	2,55	2,48	4,11
Features 3	3,11	0,14	1,77	2,28	2,35	4,54
Features 4	3,26	0,18	1,75	2,51	2,63	4,27
Features 5	3,35	0,18	1,77	2,51	2,62	4,24
Features 6	3,28	0,17	1,77	2,5	2,62	4,38
Features 7	3,24	0,18	1,77	2,45	2,61	4,23

Table 17. Standard deviation of the error using the new features

	Decision tree	Bagging regressor	Random forest regressor	k-NN	SVR	ANN
Features 8	3,22	0,15	1,66	2,79	2,79	4,56
Features 9	2,83	0,17	1,64	2,57	2,66	4,71