ANALYSIS OF PHARMACOLOGICAL DATA FOR THE PREDICTION OF THE EFFECTIVENESS OF CHEMICAL COMPOUNDS WITH CARDIO-PROTECTIVE ACTIVITY

Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής
Department of Computer Science and Engineering

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Diploma Thesis

Bakalis Dimitrios
A.M. 3033
Supervisor : Dr. Manis George

October 15, 2021

Abstract

The constant need to create better and more effective drugs, for the advancement of human health, creates new issues and challenges to scientists. Laboratory experiments, needed to be conducted on these drugs, are not only time consuming but also costly, therefore computer science is called upon to provide solutions utilizing pharmacological data analysis and machine learning. The purpose of this study is to investigate whether we can create and train a neural network to the point that it can separate and recognize different types of activity regarding new compounds as well as predict the effectiveness of specific enzymes. Additionally, we tried to single out and choose the molecular descriptors that played a key role in the training of our models. To achieve our goals we used a wide range of machine learning algorithms where they are able to perform classification as well as regression.

Keywords: drugs, enzymes, laboratory experiments, compounds, machine learning, neural networks, classification, regression.

Abstract in Greek

Η συνεχής ανάγκη για την δημιουργία καλύτερων και αποτελεσματικότερων φαρμάκων, για την βελτίωση της ανθρώπινης υγείας, δημιουργεί νέα ζητήματα και προκλήσεις για τους επιστήμονες. Τα εργαστηριακά πειράματα, που πρέπει να διεξαχθούν σε αυτά τα φάρμακα, πέραν από χρονοβόρα είναι και δαπανηρά, έτσι η επιστήμη της πληροφορικής καλείται να δώσει λύσεις χρησιμοποιώντας την ανάλυση φαρμακολογικών δεδομένων και την μηχανική μάθηση. Σκοπός αυτής της μελέτης είναι να ερευνηθεί η δυνατότητα δημιουργίας και εκπαίδεύσης ενός νευρωνικού δικτύου σε σημείο που να μπορεί να διαχωρίζει και να αναγνωρίζει τους διαφορετικούς τύπους δραστικότητας νέων ενώσεων, καθώς και να προβλέπει την αποτελεσματικότητα συγκεκριμένων ενζύμων. Επιπλέον, προσπαθήσαμε να ξεχωρίσουμε τους μοριακούς περιγραφητές των ενώσεων που είχαν καίριο ρόλο στην εκπαίδευση των μοντέλων μας. Για να επιτύχουμε τους στόχους μας χρησιμοποιήσαμε ένα ευρύ φάσμα αλγορίθμων μηχανικής μάθησης, όπου είναι σε θέση να εκτελέσουν κατηγοριοποίηση καθώς και οπισθοδρόμηση.

Λέξεις Κλειδιά: φάρμακα, ένζυμα, εργαστηριακά πειράματα, ενώσεις, μηχανική μάθηση, νευρωνικά δίκτυα, παλινδρόμηση, κατηγοριοποίηση.

Acknowledgements

Firstly, I would like to thank **Dr.George Manis** (UOI) as well as ... (UOA) for their guidance and assistance during the elaboration of my thesis. I would also like to thank my family and friends for the uninterrupted support they showed me throughout my effort.

Contents

Chapter 1

Introduction

1.1 Overview

Artificial intelligence is a computer science field that enables computers to identify, categorize and predict effectively real world data mimicking the operation of a human brain. The human brain consists of neurons that work together to form a big network that takes inputs through the five human senses and produces emotional or physical responses. A computer on the other hand, creates an artificial neural network using artificial neurons that give as a result either 0 or 1. Combining this artificial neurons we can produces results like the human brain, giving inputs to the network and producing results. The continuous development of neural networks made this field capable of detecting and labeling objects as good as a human being. Neural networks are used in the field of healthcare to help scientists with lab experiments, medical diagnosis, drug efficacy detection as well as pharmaceutical trials. Neural networks have been shown to significantly reduce the overall cost of annual medical and pharmacological expenses due to the high efficiency of the networks as well as their continuous improvement. In this study

we analyzed the enzyme lipoxygenase (LOX) as well as various new compounds of this enzyme. We used a wide range of neural network models to categorize the activity of several compounds as well as predict the actual value of the effectiveness given a specific target. We also implemented statistical models to evaluate and extract the most valuable features from each data set to make our models even more efficient.

1.2 Organization

The report is organized in the following order. Chapter 1 explains the properties of the pharmacological data that have been used in the study and introduces us to the pharmacological field that we studied. Chapter 2 introduces the neural networks that have been used to train and test the models needed for the experiments to be conducted. Chapter 3 introduces the feature selection models needed for the extraction of the best features from our data sets. Chapter 4 presents the results from the experiments performed to our data sets as well as the test sets. Finally Chapter 5 summarizes the conclusions of our study.

1.3 Problem Definition

Non-steroidal anti-inflammatory drugs are the most useful and commonly prescribed treatment for inflammatory conditions. Nevertheless, their wide use has been linked to many adverse effects, either minor (e.g., gastrointestinal irritation) or major (e.g., increased risk for cardiovascular disease). An interesting approach to overcome these risks includes the design of antioxidants compounds able to scavenge DPPH and lipoxygenase (LOX) inhibitors in parallel. The implication of reactive oxygen species in inflammatory conditions is well proven, mainly via the pro-inflammatory properties of the generated superoxide anion. The latter is involved in the deterioration of a plethora of inflammatory conditions. The problem we are called to face, using machine learning as well as data analysis is, the creation of capable machine learning models to discriminate pharmacological selectivity and discover compounds with dual activity prior to synthesis [31]. Knowledge from already in vitro tested compounds can establish an accurate filter algorithm to separate compounds with high, moderate or low affinity. We constructed a classification protocol to check whether the compounds designed could be LOX inhibitors and/or DPPH scavengers as an indication of in vitro anti-inflammatory and/or antioxidant activity, respectively. Drugs with dual activity could assist in the proper functioning of the heart and specifically in its proper perfusion. Some produced compounds of LOX enzyme oxidize the fatty acids that one needs to help with an inflammation, with the compounds belonging to the category of anti-inflammatory drugs. Additionally, when an organism is stressed it produces some free radicals therefore there are some compounds that block this radicals preventing the organism from destroying itself. This compounds

belong to the category of antioxidant drugs and we tried to categorize correctly the new compounds that have a good effect on their pharmacological target. Eventually, we wanted to observe the way in which these newly synthesized compounds are separated from each other, regarding a specific pharmacological goal, using machine learning models for classification as well as regression. Finally, the properties of each compound are being described by some molecular descriptors. This descriptors are the features of our databases, consequently we had to single out and pick the best out of them for better model training as well as using them in future studies.

1.4 Contribution

The contribution of this diploma thesis is that a systematic study and research was conducted in order to design, implement and evaluate different types of machine learning models with application to pharmaceutical data. According to the current literature, we innovated with the research and analysis conducted on the lipoxygenase (LOX) enzyme with the use of computer science. We successfully created models capable of categorizing correctly the type of activity for new synthesized compounds as well as predict the exact value of effectiveness of different proteins. Furthermore we managed to extract valuable features from the data sets in order to create new and more reliable models as well as use this features for future studies. Finally we confirmed given the results we have produced, that computer science and particularly the branch of machine learning, can play a key role in the field of health care as well as in pharmacology.

Chapter 2

Machine Learning Algorithms

2.1 Theoretical background

A neural network can be considered as a series of mathematical algorithms that tries to simulate the operation of the human brain. The human brain contains systems of neurons that communicate with each other utilizing input signals and producing output signals. A neural network mimics the operation of those neurons using artificial neurons. Interest in artificial neural networks goes back 60 years with the work of McCulloch and Pitts [15]. They introduced the "all-or-none" law of nervous activity, proposing that the activity of a neuron can be described as a mathematical proposition. A neuron has two states, either firing or not firing. In our case, a artificial neuron can be either in state 0 or state 1. In the early 50s, Rosenblatt [23] proposed the idea of the perceptron. A perceptron can be described as a mathematical model that, given a linearly separable data set the model is able to converge into a solution in finite number of steps. Despite the fact that the perceptron seemed promising, it had major flaws. A single layer perceptron was unable to solve multi class problems as it was impossible to learn the XOR

function. In this regard, Hopfield pointed out in his study [11] that the perceptron had critical limitations in non-linear problem solving. He proposed the idea of feedback, meaning the backward coupling of outputs with inputs to tackle this problem. The need to improve the perceptron model, had brought to light the idea of using multiple layers of perceptrons to create a more capable model. Working with the concept of multilayer perceptron Hinton, Rumelhart and Williams in their study [24] proposed the generalized delta rule. Using this method, the weights of the neurons are updated through back-propagation for a more effective training of the model. Over the years, the delta rule has shown to be a great success in problem solving and the idea of multilayer perceptron-like machines has been established in modern machine learning. The advancement of neural networks has brought computer science closer to other sciences such as chemistry, physics and medicine. Specifically in medicine, neural networks thrived in the field of pharmaceutical research combining data analysis and machine learning as it has been proposed by a study in 2013 [28].

2.2 Perceptron

Perceptron is a mathematical model suitable for classification problems. We give input data at the perceptron and it calculates the output. The calculations are made by, multiplying the input vector $x = (x_1, x_2, ...x_n)$ with the weight vector $w = (w_1, w_2, ..., w_n)$ and comparing the sum of all the multiplied values to a threshold θ using an appropriate activation function. All of that can be expressed as a formula [1] f of the form

$$f(x) = sgn(w \cdot x - \theta).$$

The multiplication detonates the inner product $\sum_{i=1}^{n} w_i x_i$ and if the value of the product is equal or greater than the threshold, the neuron fires giving 1 as a result. In any other case, the neuron doesn't fire giving the value 0 as a result.

$$sgn(a) = \begin{cases} 1 & \text{if a} \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

The mathematical being of the perceptron is shown in Figure 3.2

Figure 2.1: Correlation between a Neuron and a Perceptron

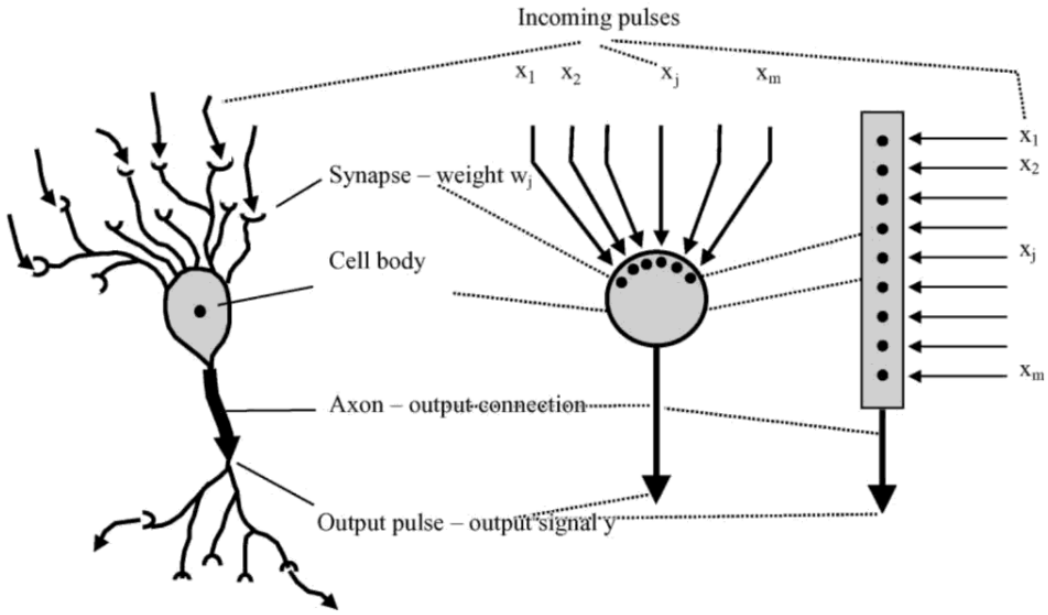
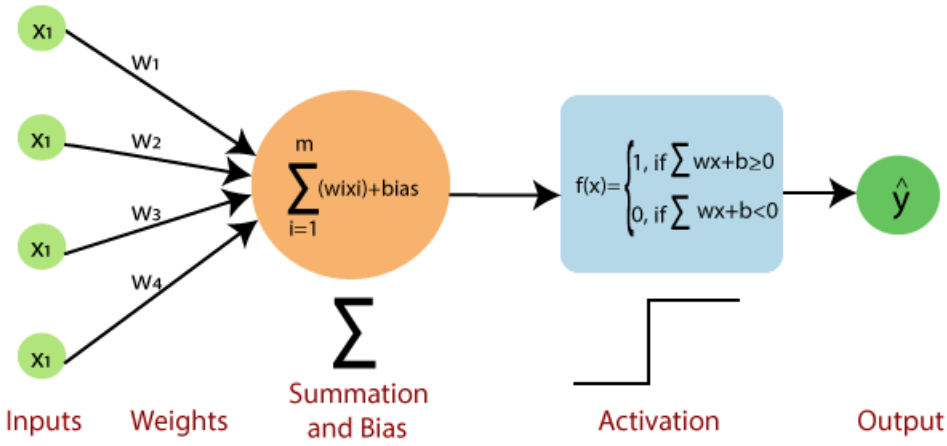


Figure 2.2: Mathematical model of a Perceptron



2.3 MLP

A multilayer perceptron is considered to be a feedforward network as shown in Figure 3.3, meaning that there is no output feedback of a neuron to the neurons of which directly or indirectly affected. Neurons are organized into layers, being no connections between neurons of the same level. A multilayer perceptron consists of a input layer, a number of middle hidden layers and the output layer. Except for the input nodes, every other node is considered to be a neuron having a non-linear activation function. A MLP with at least one hidden layer with nonlinear neurons can approach any non-linear function, sufficiently increasing the number of hidden neurons as shown in Figure 3.4. In order to make the MLP more effective and accurate we train it using the backpropagation algorithm.

Figure 2.3: Feed forward network

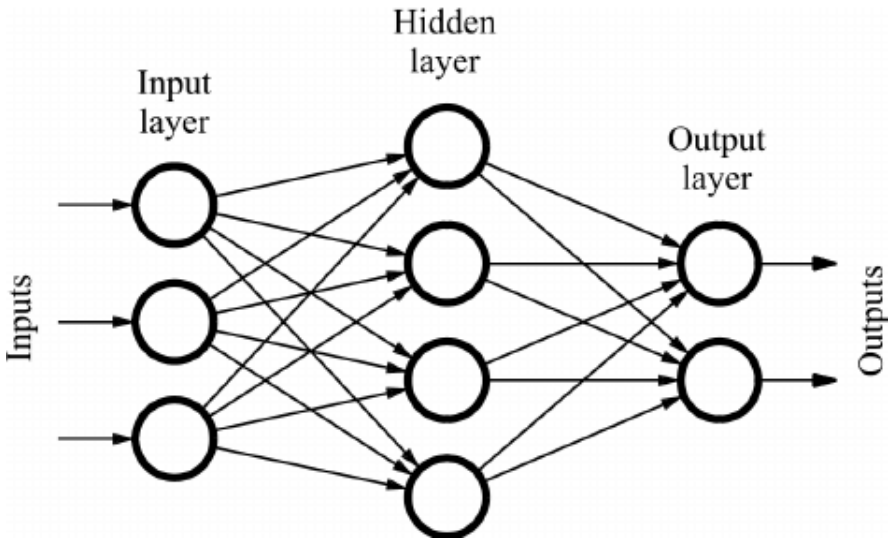
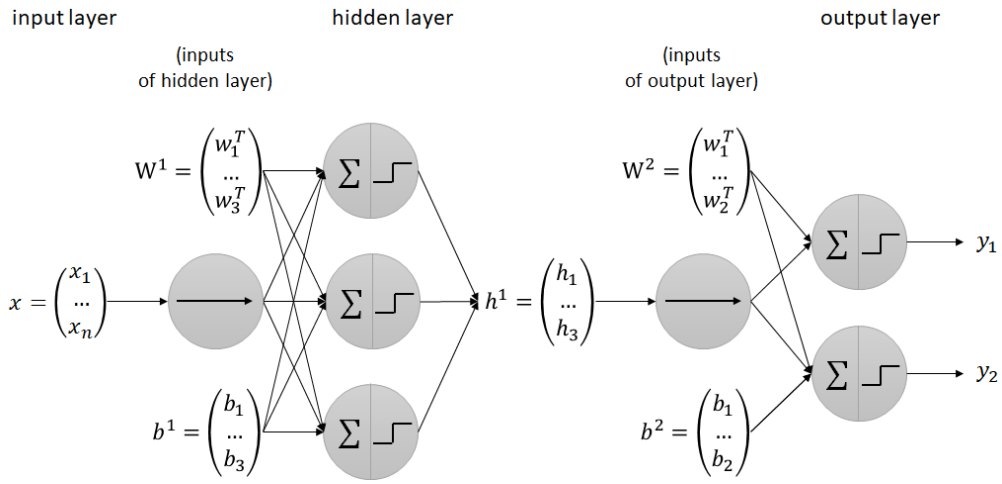


Figure 2.4: MLP with one hidden layer



Backpropagation is a method where the weight and bias of every neuron are updated in order to minimize the error, usually referring to root mean square error. The algorithm works with a forward pass and a backward pass. In forward pass, the information flows from the input layer to the output layer calculating the output of each neuron of the network. The results are cross validated with the ground truth labels to check the performance of the network. In backward pass, the final network outputs are compared to the desired ones, producing the error in the MLP outputs. The error signals are propagated backwards in the network and the error for the neurons of each level is gradually calculated from the last hidden level to the first. The error of the output node can be expressed as a formula of the form

$$\delta_j(n) = t(n) - y(n),$$

where t is the desired value and y is the produced value from

the network with n being the n_{th} data point. The total error of the network is given as a formula of the form

$$E(n) = \frac{1}{2}\sum_{j}\delta_j(n)^2,$$

and it is called Mean Square Error with j being every neuron of our model. To minimize the error of the output we use gradient descend, adjusting the the neuron weights as shown in the formula [9] below

$$\Delta w_{ij}(n) = -r\frac{\partial E(n)}{\partial u_j(n)}y_i(n),$$

with r being the learning rate and y_i being the output of the previous node. The change of weight for a specific node depends of the activation function of this node. The activation functions of the MLP are sigmoimds with the most common being : the logistic function described by Equation 3.2 ranging from 0 to 1 as shown in Figure 3.5, the hyperbolic tangent described by Equation 3.3 ranging from -1 to 1 as shown in Figure 3.6 and the rectifier liner unit described by Equation 3.4 that has been proposed recently in machine learning as shown in Figure 3.7. The formulas for the activation functions are described below

$$y(u_i) = (1 + e^{-u_i})^{-1}, \tag{2.2}$$

$$y(u_i) = tanh(u_i), \tag{2.3}$$

$$f(x) = x^+ = max(0, x). \tag{2.4}$$

Figure 2.5: Logistic Activation Function

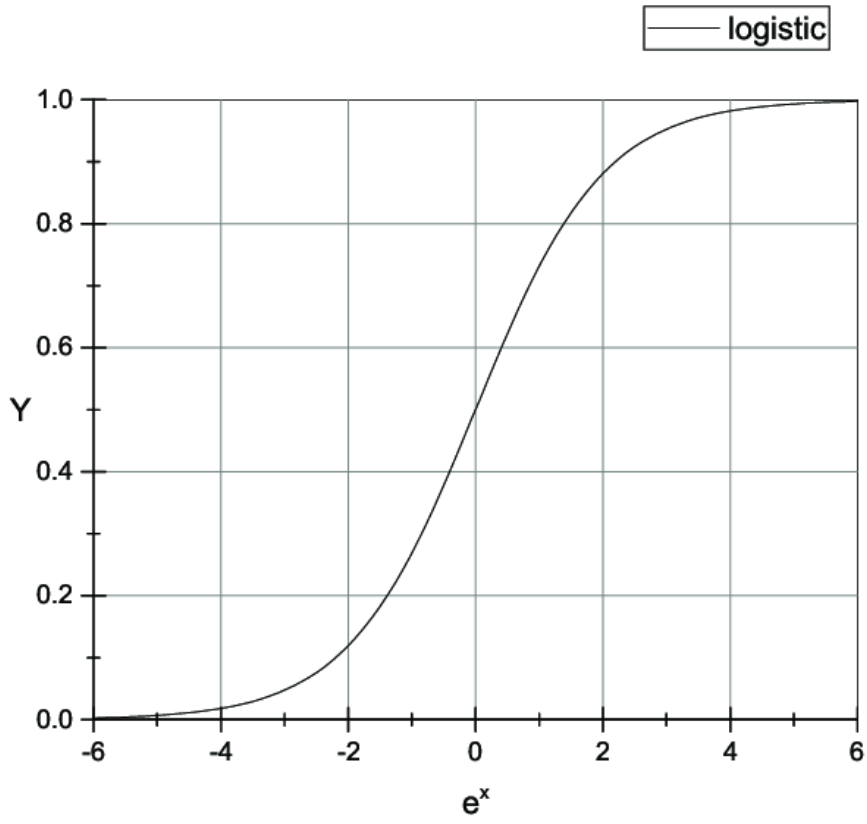


Figure 2.6: Hyperbolic Tangent Activation Function

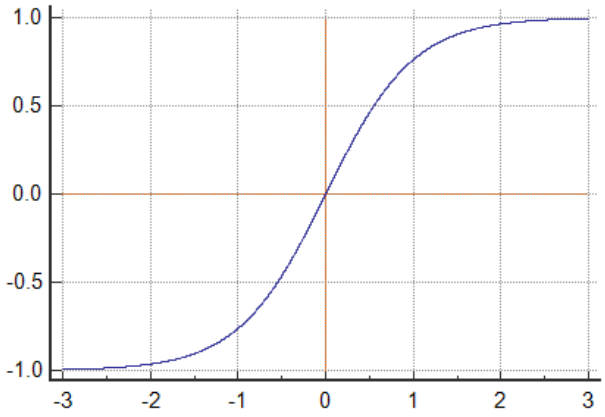
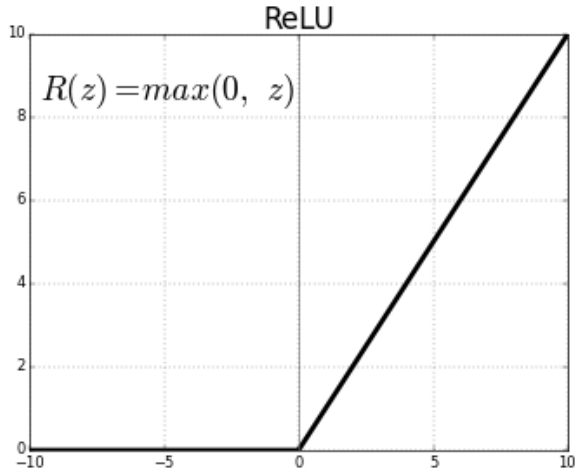


Figure 2.7: ReLU Activation Function



2.4 Logistic Regression

Logistic Regression is a machine learning algorithm suitable for binary classification problems as well as multiclass problems. The main feature of the this algorithm is the use of the logistic function. The algorithm is based on probability predictions using a complex cost function, defined as the Sigmoind function. The expectation hypothesis limits the function between 0 and 1 making linear problem solving impossible as there can be values greater than 1 and less than 0.

$$0 \leq h_\theta(x) \leq 1$$

The logistic regression hypothesis [10] can be expressed as a formula of the form

$$h_\theta(X) = \frac{1}{1 + e^{-(\beta_o + \sum \beta_i X_i)}}$$

The way that the model works is that, we give the model a set of inputs and it passes them through the prediction function in order to get a value between 0 and 1. The class that an input belongs to is defined by the value of the prediction compared to a predefined threshold. If the value is greater than the threshold then the input belongs to class A in any other case it belongs to class B. In order to make the model more accurate, a cost function [4] has been created representing an optimization objective as it can be seen in Figure 3.8 and 3.9. The cost function can be described as a formula of the form.

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if y} = 1 \\ -\log(1 - h_\theta(x)) & \text{if y} = 0 \end{cases} \tag{2.5}$$

Figure 2.8: Cost Function for y = 0

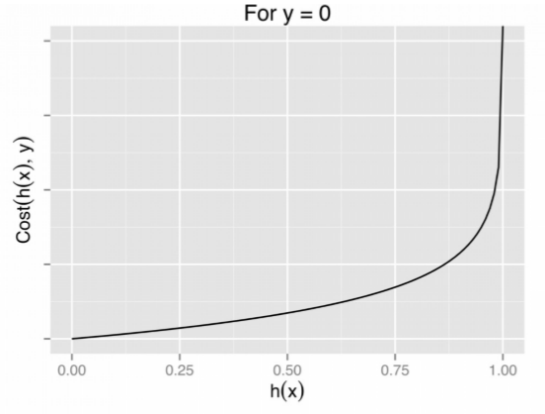
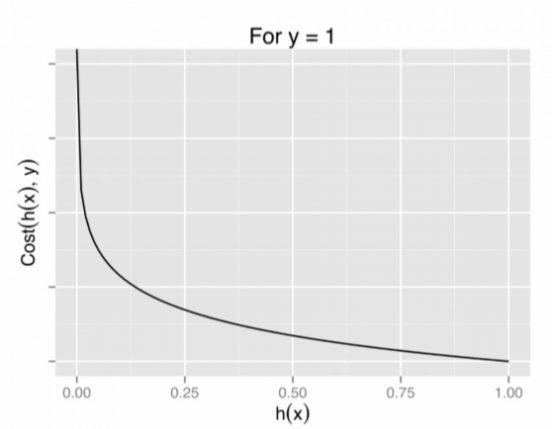


Figure 2.9: Cost Function for y = 1



We can create a unified equation to express the two formulas that describe the cost function. The equation is described below

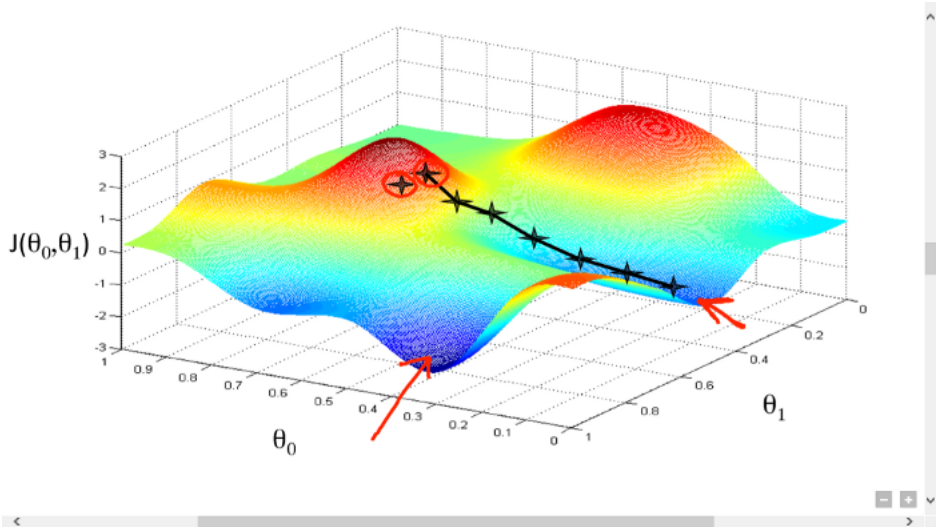
$$J(\theta) = -\frac{1}{m}\sum[y^{(i)}\log(h_\theta(x(i))) + (1 - y^{(i)}\log(1 - h_\theta(x(i)))]$$

Like the MLP model, we want to reduce the cost value. The way to do that is, by using the Gradient Descent method to minimize the cost value. The method is applied on each parameter of our model.

$$\theta_j := \theta_j - a\frac{\partial}{\partial\theta_j}J(\theta)$$

$$\theta_j := \theta_j - a\sum_{i=1}^{m}(h_\theta(x^{(i)} - y^{(i)})x_j^{(i)})$$

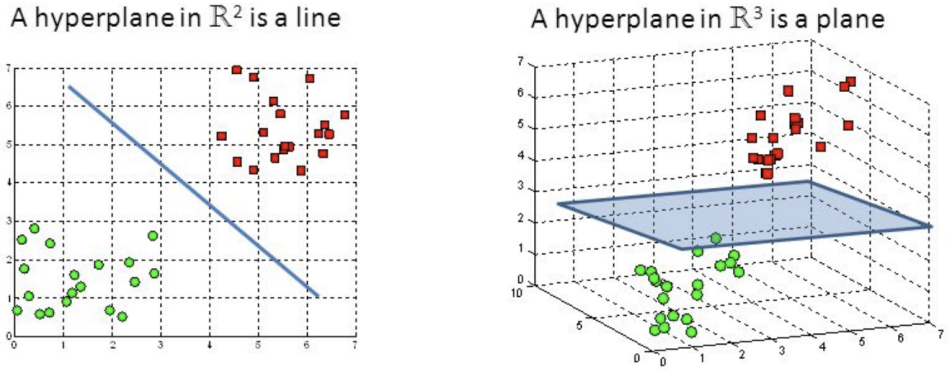
Figure 2.10: Gradient Descent 3-D visualization



2.5 Support Vector Machine

Support Vector Machine also known as SVM, is another machine learning algorithm for solving classification problems. The way the algorithm works is by defining a hyperplane as shown in Figure 3.11 to classify the input set. The inputs are represented as data points with the number of dimensions being the number of classes we want to use.

Figure 2.11: Hyperplane in 2-D and 3-D



A hyperplane can be thought of as a threshold for the data points. The goal of the algorithm is to find the best hyperplane that fits the data with the maximum margin. In case of data points, we want a hyperplane that has the maximum distance between data points to achieve a accurate prediction. In order to make the model more accurate we need to maximize the margin of the distances for the classification of new input sets. To maximize the margin, the algorithm uses some key points called support vectors essential for building the model as shown in Figure 3.12. This vectors affect the position as well as the

orientation of the hyperplane and removing them may cause the hyperplane to change. As with the Logistic Regression, a loss function has been introduced [29] called the hinge loss, to help the algorithm maximize the margin. The function can be described as a formula of the form.

$$c(x, y, f(x)) = \begin{cases} 0 & \text{if y*f(x)} \geq 1 \\ 1 - y * f(x) & \text{else} \end{cases} \tag{2.6}$$

The above formula tells us that we get a value of 0 if the predicted value is equal to the true value. In any other case the loss value is calculated but, it is imbalanced. To balance the maximization of the margin and the loss, a regularization parameter is needed to be added to the calculation of the cost function. Adding this new parameter, the cost function formula [29] can be described as.

$$min_w \lambda \|w\|^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

As in every machine learning algorithm, the weights of the network have to be updated to achieve a more accurate model. We calculate the partial derivatives to find the gradients.

$$\frac{\partial}{\partial w_k} \lambda \|w\|^2 = 2\lambda w_k$$

$$\frac{\partial}{\partial w_k} (1 - y_i \langle x_i, w \rangle)_+ = \begin{cases} 0 & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik} & \text{else} \end{cases} \tag{2.7}$$

The way we update the weights depends on whether the model predicted the label of the input correctly or not. So we have two

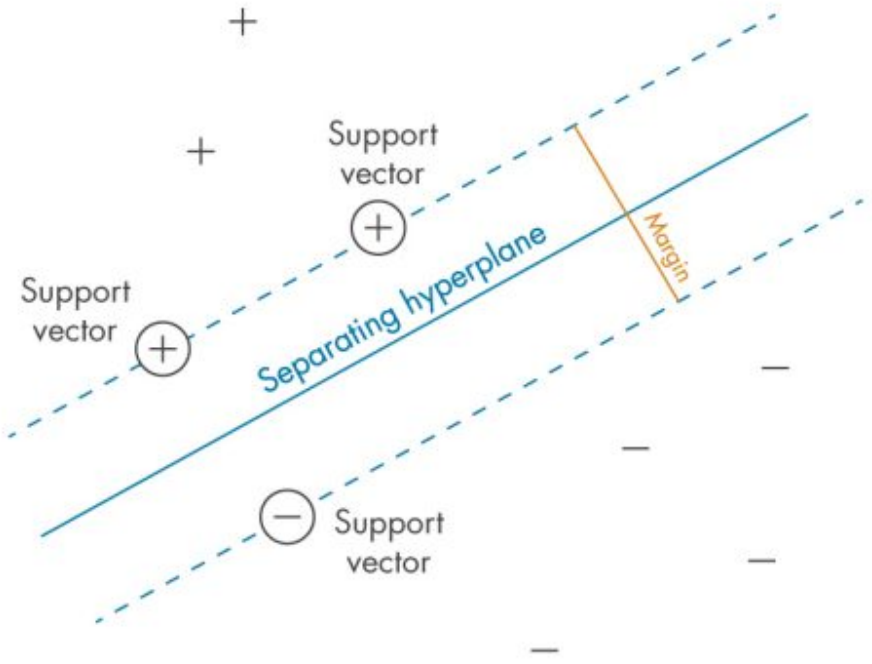
ways on updating the gradient. If the model predicted the label correctly then we update the gradient from the regularization parameter.

$$w = w - a \cdot (2\lambda w)$$

If the model predicted the label incorrectly then we update the gradient by including the loss together to the regularization parameter.

$$w = w + a \cdot (y_i \cdot x_i - 2\lambda w)$$

Figure 2.12: SVM



2.6 Quadratic Discriminant Analysis

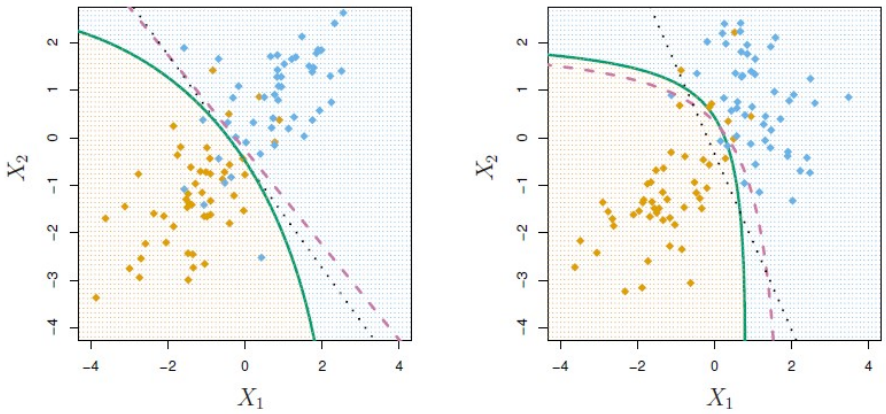
Quadratic Discriminant Analysis or QDA is a statistical classifier based on quadratic discrimination for separating features for two or more classes. In QDA we do not know whether the covariance matrix of each class is identical to the others. So we will calculate the covariance matrix Σ_κ separately for each class k. The Quadratic discriminant function [6] can be described by the formula of the form:

$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T\sum_k^{-1}(x-\mu_k) + \log\pi_k.$$

For the classification rule, the model has to maximize the the quadratic discriminant function with k classes.

$$\hat{G}(x) = \arg\max_k \delta_k(x)$$

Figure 2.13: Decision boundaries of QDA method



2.7 K-Nearest Neighbors

K-Nearest Neighbors [12] is another machine learning algorithm, suitable for solving both regression and classification problems. The way that the model work is by, representing the data as 2-D vectors on a graph and trying to label similar points to a specific class by calculating the distance between this points. The number of points tested is defined the by the k variable. The idea of the model with "k=1" can be seen in Figure 3.13. For measuring the distance between the points, the method uses the Euclidean distance [2] described by the formula:

$$d(p, q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2},$$

with p and q being the two points, p_i and q_i being the Euclidean vector starting form the origin of the space and n being the dimensions of the plane. The Nearest Neighbors algorithm works best when the input data has two features. If the are more than two features, a dimension reduction algorithm is applied to the data to reduce the number of features to "two". A well known algorithm capable for this kind of work is called Principal Component Analysis or PCA [13]. PCA is visualized in figure 3.15.

Figure 2.14: Nearest Neighbors with k=1

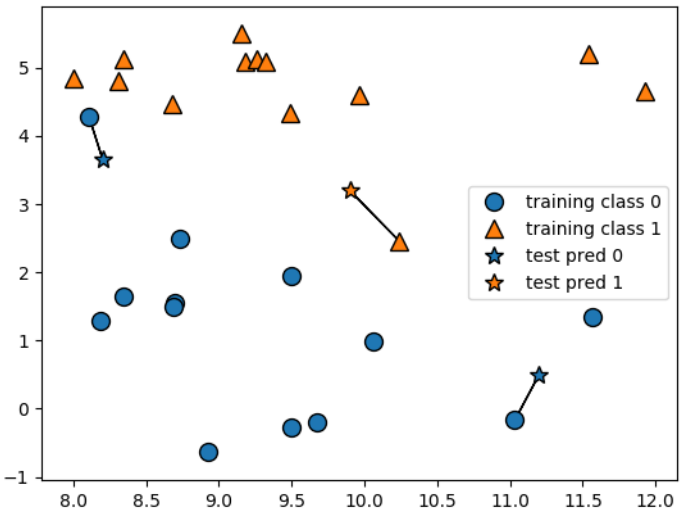
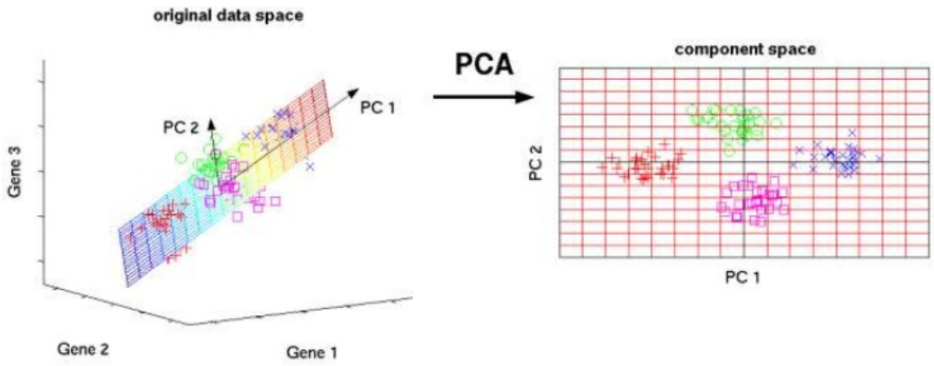


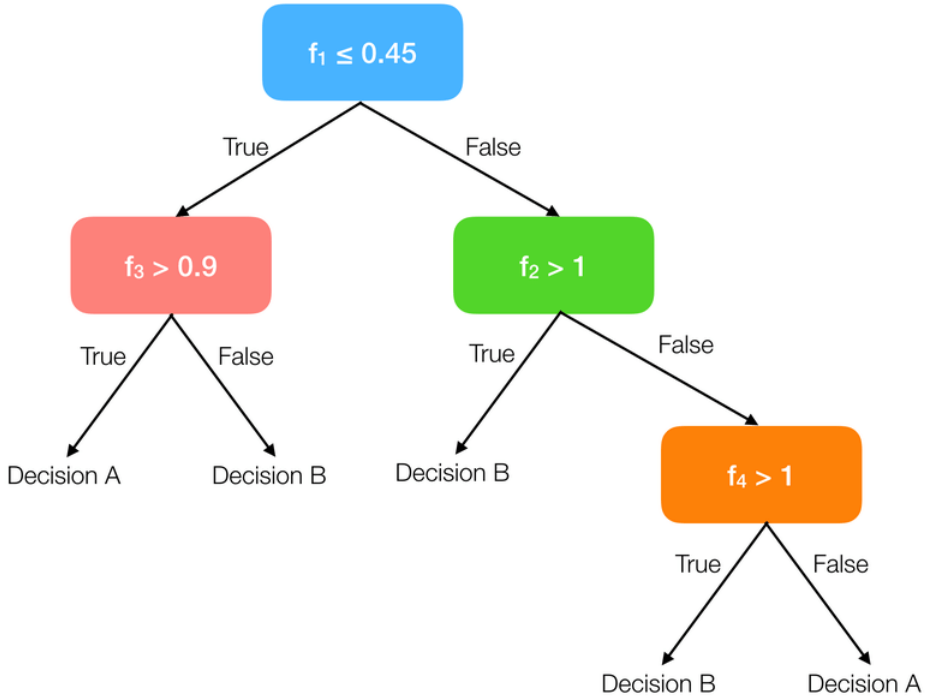
Figure 2.15: Principal Component Analysis



2.8 Decision Tree

Decision Tree classifier mimics the structure of a real world tree. A tree has a root, internal nodes or branches and the leafs or terminal nodes. The way that the Decision Tree works is by inserting the input data set on the root node, then testing the attributes on every internal node and then assign the input on a leaf node as the output. Leaf nodes are assigned with the different classes of the problem the algorithm tries to tackle. Every non-terminal node contains a test condition for the features to be tested. The representation of a Decision Tree can be seen in Figure 3.16.

Figure 2.16: Decision Tree



The training of a Decision Tree model depends on the conditions of the non-terminal nodes. The algorithm splits the attributes to test them in the nodes and decides whether it was a good split for each individual class. The training data set is getting split several times until all the subsets belong to the same class. Several algorithms can be used to measure the best metric for each split and the final quality of the split comes from the average value. These algorithms can be thought of as criteria for a single split. Such an algorithm is the Information Gain, based on the idea of entropy and information. The algorithm tries to find the best feature to make the split for each node when building the tree. Entropy can be described as a formula of the form:

$$H(T) = I_E(p_1, p_2, ..., p_J) = -\sum_{i=1}^{J} p_i \log_2 p_i$$

Information Gain can be described [18] as the entropy of the parent node subtracting the sum of entropy of the children nodes.

$$IG(T, a) = H(T) - H(T|a)$$

$$IG(T, a) = -\sum_{i=1}^{J} p_i \log_2 p_i - \sum_{i=1}^{J} iPr(i|a) \log_2 Pr(i|a)$$

The Expected Information Gain can be described the entropy of the parent node subtracting the weighted sum of entropy of the children nodes.

$$IG(T, a) = H(T) - H(T|A)$$

$$IG(T,a) = -\sum_{i=1}^{J} p_i \log_2 p_i \sum_{a} p(a) - \sum_{i=1}^{J} iPr(i|a) \log_2 Pr(i|a)$$

Another metric for splitting the data is Gini Index [18]. It is based on the Tsallis Entropy [14] being a variation of the entropy we saw above. Gini impurity can be given by the formula below

$$IG(p) = \sum_{i=1}^{J} (p_i \sum_{k \neq i} p_k) = 1 - \sum_{i=1}^{J} {p_i}^2,$$

with J being the number of classes and p_i being the fraction of labeled items in class i.

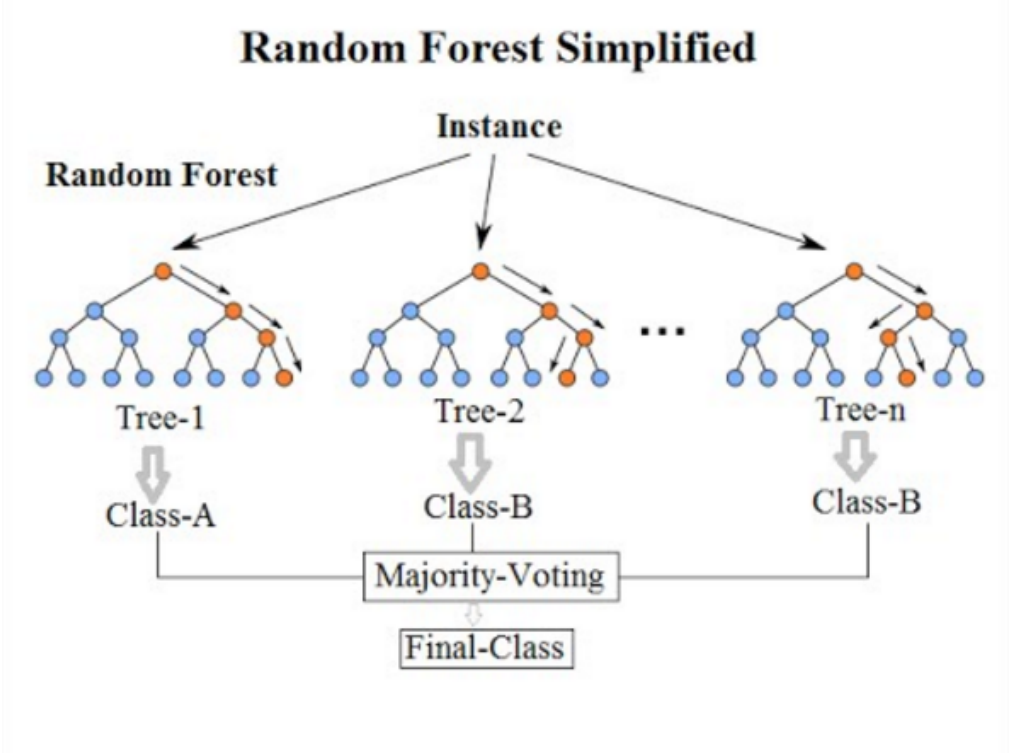
2.9 Random Forest

The Random Forest classifier is based on the idea of the Decision Tree. They way the model works is by combining multiple individual Decision Trees that each one of them makes a prediction. The model decides the class of the output by choosing the one with the most votes. The Random Forest algorithm is more reliable compared to the Decision Tree because the errors, that may occur, in some of the inferior trees doesn't affect other trees of the same group nor the final output. As discussed in the Decision Tree section, the Random Forest algorithm uses the Gini Index [16] as an feature selection method. The method can be described as a formula of the form:

$$\sum \sum_{j \neq i} (f(C_i, T)/|T|)(f(C_j, T)/|T|),$$

where T is the training set, C_i is a class and $f(C_i, t)/|T|$ being the probability that the selected input belongs to class C_i. With this way of operation the model avoids the problem of overfitting and is more reliable as it sets many test conditions for the features in the inferior trees rather than using just one tree. The way that the model is constructed can be seen in Figure 3.17

Figure 2.17: Random Forest



2.10 Naive Bayes

Naive Bayes is another model used for classification problems. The whole idea of the model revolves around the Bayes Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

This equation gives us the probability of A occurring, given a B statement. A can be thought of as the hypothesis and B as the true evidence. The cases have to be independent hence the name naive. If we want to express the formula [22] using real training data we can write it as:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)},$$

with y being the class label and $X = (x_1, x_2, ..., x_n)$ being the features of the data set. This formula tackles problems with two classes, most of the time problems with answers of the form "yes" or "no". We can express the probability for the whole data set as a formula:

$$P(y|x_1, x_2, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y).$$

If we want to solve problems with multiple classes the above formula can be given by h [22], calculating the maximum probability:

$$h = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i|y).$$

2.11 AdaBoost

Adaptive Boost, also known as AdaBoost [8], is an algorithm that combines weaker machine learning models to achieve better accuracy scores. The main idea of the model can be expressed as a formula of the form:

$$F_T(x) = \sum_{t=1}^{T} f_t(x),$$

where f_t is a weaker algorithm that takes an input of x and returns a class label. We have a training set of $(x_1, c_1), ..., (x_n, c_n)$ with x_i being the input and c_i being a number that represents a class out of a finite set $(1, 2, ..., K)$. K is the number of classes in the data set. The goal of the algorithm is to find a $Z(x)$ classification rule so that it can assign correctly a label to a new input x from $(1, ..., K)$. The error rate of misclasiffication can be expressed by a formula of the form:

$$Z(x) = 1 - \sum_{k=1}^{K} E_X[I_{C(X)=k} Prob(C = k|X = x)]$$

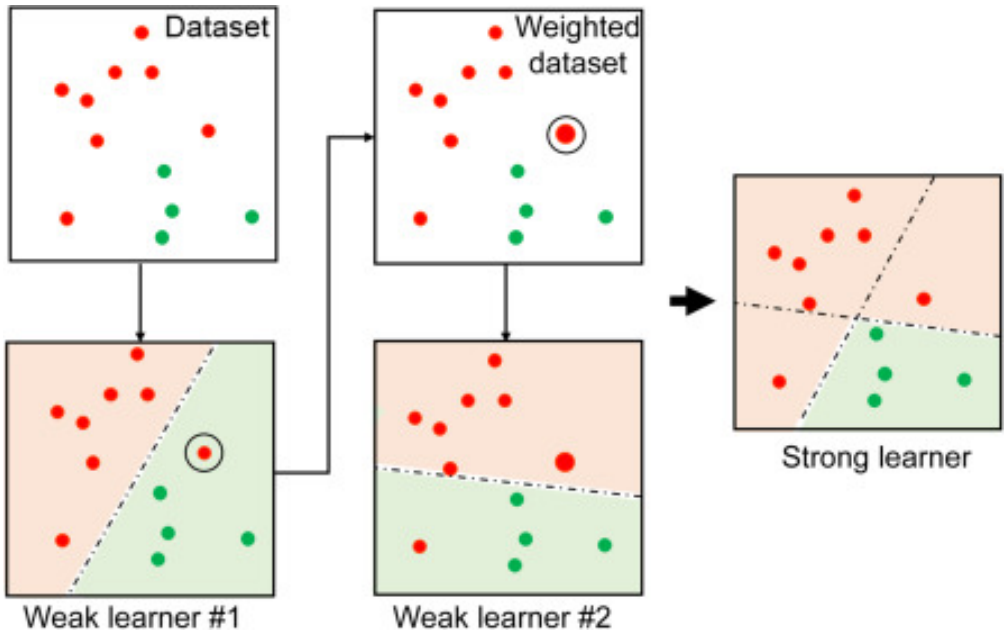
The algorithm tries to reduce the error rate to a minimum value using :

$$Z^*(x) = \arg\max_k Prob(C = k|X = x),$$

The model starts with an unweighted training set and makes a prediction for an input x. If the prediction of the label is wrong, then the weight of the input is increased (boosted). Then the model picks another algorithm and runs the same procedure

with the boosted weights and so on. With this method the algorithm could build over 1000 internal models to find the best solution combining the outcomes of this models. AdaBoost has shown to be extremely successful when used in two-class classification problems. The training processes of the model can be seen in Figure 3.18.

Figure 2.18: Adaptive Boost



2.12 Gradient Boosting

Gradient Boosting [5] is yet another machine learning algorithm suitable for solving both classification as well as regression problems. The main idea of the algorithm is by combining weaker learning models, most commonly Decision Trees, to produce a gradient boosted tree. The goal of the algorithm is to approximate the output given a set of input variables with a function of the form $\hat{F}(x)$. The algorithm is applied in supervised learning problems and the way that the model updates itself is by trying to reduce the loss function $L(y, F(x))$ to a minimum. The way of minimization can be expressed as a formula of the form :

$$\hat{F} = \arg\min_{F} E_{x,y}[L(y, F(x))],$$

where x is the input variables and y is the output variable. The algorithm handles a training data set, given the ground truth y, $(x_1, y_1), ...(x_n, y_n)$, with n being the number of input variables, calculating the approximation function $\hat{F}_n(x)$ in each iteration to minimize the average loss. It starts by calculating the first approximation $\hat{F}_0(x)$ and continues until the last input variable of the data set, combining in each iteration the weaker models :

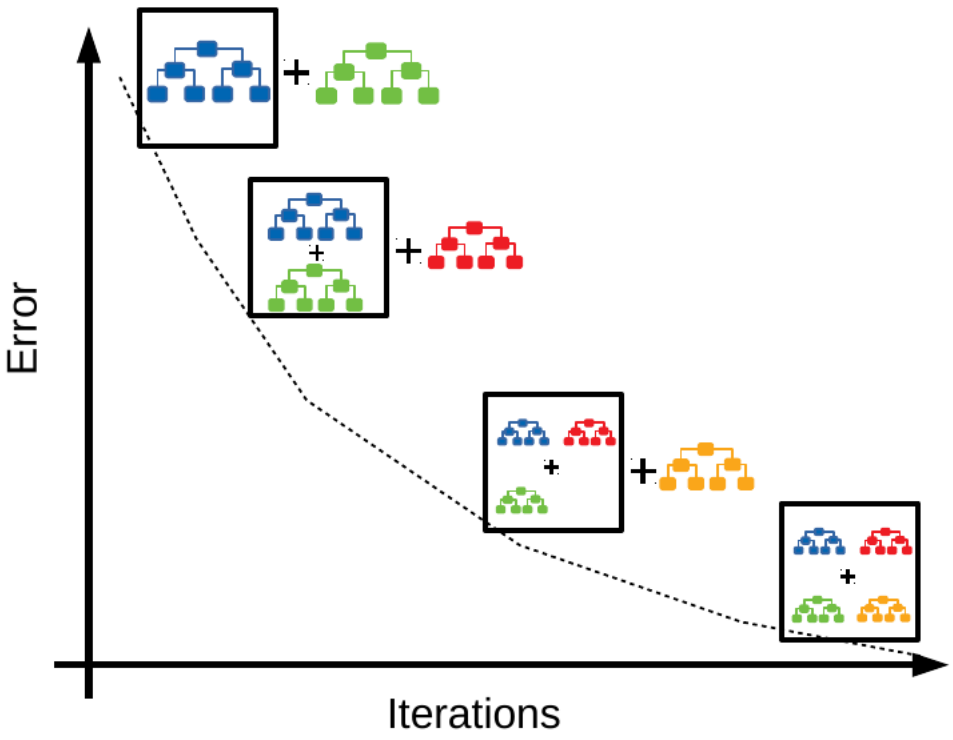
$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma),$$

$$F_m(x) = F_{m-1}(x) + \arg\min_{h_m \in H} \left[\sum_{i=1}^{n} L(y_i, F_{m-1}(x) + h_m(x_i)) \right],$$

with $h_m \in H$ being a weaker learner.

The concept of the Gradient Boosting algorithm as well as the loss function minimization can be seen in Figure 3.19 below

Figure 2.19: Gradient Boosting Algorithm



Chapter 3

Feature Evaluation

3.1 Introduction

Feature evaluation methods are widely used from machine learning engineers to help them build more efficient and reliable models. A feature can be thought of as a value for a specific property of a subject. In machine learning we often have data sets that consists of multiple features thus making the training of a single model complex. Feature selection methods are mathematical models that are applied in the data set to highlight the best features depending on the method we use. The use of this models is essential in building faster and more robust neural networks.

3.2 Information Gain

Information Gain also known as "Info-Gain" is one of the most popular, feature selection method in machine learning. The way that the method works is by calculating the entropy of the dataset S in regard to the entropy of the dataset given a specific feature a. The method can be expressed as a formula [30] of the form:

$$IG(S, a) = H(S) - H(S|a).$$

3.3 Chi-Squared

Chi-square is a statistical method used in machine learning for selecting the best features from a data set. A feature is a real number and it can follow the chi-square distribution if it can be written as a sum of squared normal variables $x^2 = \sum Z_i^2$. In machine learning we use the chi-squared test to calculate the dependency between two events. To conduct the test we use a formula [26] called chi-square formula :

$$x_c^2 = \sum \frac{(O_i - E_i)^2}{E_i},$$

where O is the observed value and E is the expected value. If we get a low value out of our formula it means that, the observed value is close to the expected value thus the two features are independent. If we get a high chi-square value, we have strong dependency meaning that the features are more suitable for our needs.

3.4 Gini Index

Gini Index is another feature selection method used in machine learning and specifically in decision tree algorithms. The way that the algorithm works is [17] by using a data set S with s samples and n different classes $(C_i, i = 1, .., n)$. It divides the set of samples into n subsets $(S_i, i = 1, .., n)$. The Gini Index of the set S can be calculated by the given formula:

$$Gini(S) = 1 - \sum_{i=1}^{m} P_i^2,$$

where P_i is the probability that any sample s_i/s belongs to C_i. A new and improved Gini Index was proposed in 2007 [27], using the purity of the features. The higher the purity value, the more significant the attribute is. The new formula can be described as:

$$Gini(W) = \sum_{i}^{n} P(W|C_i)^2 P(C_i|W)^2,$$

where W is a feature and C_i the i-th class.

3.5 Pearson Correlation Coefficient

Pearson Correlation Coefficient measures the relationship between two features. It is an effective feature selection method used in machine learning problems. The way that the method works is by calculating the correlation between two variables x and y. The method can be expressed as a formula [25] of the form:

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

where n is the sample size, x_i and y_i being the features we want to measure with $\bar{x}$ and $\bar{y}$ being the sample mean ($\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, same for $\bar{y}$).

3.6 Fisher Ratio

Fisher Ratio analysis can be considered as a Signal-to-Noise ratio measurement between two features. We need to keep the features that has lesser "noise" than the others. The method is used to calculate the ratio of each individual feature and then we select the ones with the highest score. The way that the algorithm works is by calculating the ratio of the variance between classes in regard to the variance of within classes. The method can be expressed as a formula [3] of the form:

$$d = \frac{(w \cdot \mu_1 - w \cdot \mu_2)^2}{w^T \sum_1 w + w^T \sum_2 w},$$

with μ_1, μ_2 being the mean vectors and $\sum_1, \sum_2$ being the covariances.

Chapter 4

Experiments

4.1 Library

For this study we used Scikit-learn [20], which is a Python based module that implements a wide range of machine learning algorithms. This library was developed by a team of researchers from various universities in 2011 and it is free under the BSD license. The benefits of Scikit-learn are that, it has build in neural networks that use NumPy [7] arrays for the data and it lets us to change the parameters of each neural network we want to use. As far as the study is concerned, we trained our models in a AMD Ryzen 5 1600 Six-Core Processor (CPU) with 16GB DDR4 (RAM) and a AMD RX 580 8GB GDDR5 (GPU).

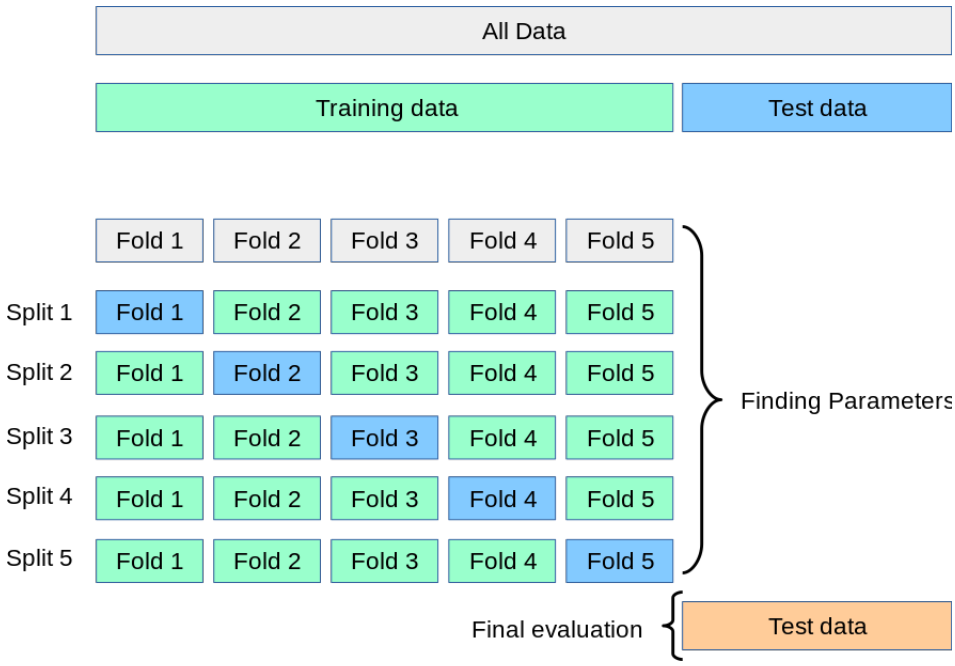
4.2 Datasets

In this study we used three pharmacological data sets. Each data set contains a target and numerical values, also known as descriptors, that describe different compounds of a specific enzyme. The first data set has 673 compounds with 243 features and it is split into three categories. Category values are 1,2 and 3, each describing the effectiveness of a compound. The second data set has 9779 compounds with 277 features and it is split into 4 categories, with value 1 showing the anti-inflammatory activity of a compound, value 2 showing the antioxidant activity of a compound and values 3 and 4 being decoy compounds. The third and final data set has 8182 compounds with 284 features. The target is in the form of decimal values, rather than categories, that shows the exact value of effectiveness for a specific type of enzyme. The first and second data sets were used for classification and the third data set was used for regression. To test our models we used two sets of 39 and 24 compounds respectively, that were composed and tested in the laboratory of the Department of Pharmacy of Athens. The first test set is used to evaluate the ability of our models to classify correctly the different types of activity and the second test set is used to evaluate the regression ability of our models.

4.3 Model Training and Cross-Validation

To train our models we used 70 % of our data and for testing the accuracy of each model the remaining 30 %. We applied the 10-k fold cross-validation method, from the Scikit-learn library, to every train set so that we can avoid overfitting our models. Cross-validation [21] is a method were the training set is split into k smaller sets. For every smaller set, being produced, the model is trained and the remaining sets are used to test the accuracy of the model. Upon completion of the method the average score of those splitted sets gives us the final train accuracy. However a small portion of the initial data set should be kept for our final evaluation, as shown in Figure 5.1.

Figure 4.1: 5-k Cross-Validation



4.4 Presentation and Analysis of Results

Concerning the first data set we achieved a maximum accuracy of 64.3% but we expected no more than that, since the data set was not suitable for classification in the first place. We started by training 10 models, for classification, using 70% of the data and the remaining 30% we used them for training. The results are shown in the table below.

Model	Train Accuracy	**Test Accuracy**	Completion Time
Logistic Regression	56.89%	64.35%	0.637 sec
MLP	58.82%	63.86%	10.024 sec
SVM (RBF)	59.23%	60.89%	0.508 sec
Random Forest	63.08%	59.90%	3.725 sec
AdaBoost	59.64%	58.91%	3.753 sec
Gradient Boosting	60.29%	57.92%	47.394 sec
Nearest Neighbors	59.24%	57.42%	0.225 sec
Naive Bayes	54.35%	54.45%	0.023 sec
QDA	56.05%	53.46%	0.682 sec
Decision Tree	51.84%	51.48%	0.567 sec

We scaled our data from values 0-1, using the MinMaxScaler normalization method [19], to try and increase the accuracy of our models. The results are shown in the table below:

Model	Train Accuracy	**Test Accuracy**	Completion Time
Logistic Regression	59.86%	64.35%	0.662 sec
MLP	60.72%	63.86%	9.611 sec
Random Forest	61.81%	62.87%	3.514 sec
AdaBoost	59.64%	58.91%	3.775 sec
Nearest Neighbors	59.45%	57.92%	0.240 sec
SVM (RBF)	60.72%	59.92%	0.489 sec
Gradient Boosting	59.87%	59.90%	47.361 sec
Naive Bayes	53.50%	54.45%	0.021 sec
Decision Tree	52.89%	53.46%	0.570 sec
QDA	56.26%	51.98%	0.707 sec

Then we applied the methods on the first data set and we extracted the top 15 best features starting with the feature that had the highest score and presenting them in descending order. The results are given in the table below:

Evaluator	Feature Ranking
Chi-Square	r_desc_Lopping_centric, r_desc_Balaban_centric r_desc_Lopping_centric, r_desc_Total_structure_connectivity, i_desc_Sum_of_topological_distances_between_Br..Br, i_desc_Sum_of_topological_distances_between_Br..I, i_desc_Sum_of_topological_distances_between_N..I, r_desc_PEOE14, i_qp_#acid, i_qp_RuleOfThree, i_desc_Chirality_count, i_desc_Sum_of_topological_distances_between_O..O, r_desc_MR1, i_qp_#stars, i_qp_RuleOfFive, i_qp_#rtvFG
Info Gain	r_qp_volume, r_qp_SASA, r_qp_glob, r_qp_PSA, r_qp_QPPCaco, r_qp_FISA, r_desc_Average_valence_connectivity_index_chi5, r_desc_Balaban-type_index_from_Z_weighted_distance_matrix__Barysz_matrix, r_desc_Balabantype_index_from_electronegativity_weighted_distance_matrix, r_desc_Balabantype_index_from_mass_weighted_distance_matrix, r_desc_Balabantype_index_from_polarizability_weighted_distance_matrix, r_desc_Balabantype_index_from_van_der_waals_weighted_distance_matrix, r_desc_Estate_topological_parameter, r_desc_Valence_connectivity_index_chi-5, r_desc_Average_valence_connectivity_index_chi3
Gini Index	r_desc_Lopping_centric, r_desc_Balaban_centric, i_desc_Sum_of_topological_distances_between_Br..Br, i_desc_Sum_of_topological_distances_between_Br..I, i_desc_Sum_of_topological_distances_between_N..I, i_desc_Sum_of_topological_distances_between_F..Br, r_desc_ALOGP8, r_desc_Total_structure_connectivity, r_desc_MR1, i_qp_#acid, i_desc_Sum_of_topological_distances_between_O..Br, i_desc_Ring_Count_8, i_desc_Sum_of_topological_distances_between_F..Cl , r_desc_PEOE14 , i_desc_Ring_Count_4
Pearson Correlation Coefficient	r_desc_Narumi_Geometric_Topological, r_desc_Narumi_Harmonic_Topological, r_desc_Molecule_cyclized_degree, i_desc_Cyclomatic_number, r_desc_Spanning_tree_number, i_desc_Total_ring_size, i_desc_Bonds_in_Ring_System, i_desc_Atoms_in_Ring_System, i_qp_#ringatoms, i_desc_Ring_perimeter, i_desc_Number_of_ring_systems, i_qp_#in56, r_desc_PEOE3, r_desc_path/walk_5__Randic_shape_index, r_desc_path/walk_4__Randic_shape_index
Fisher Ratio	i_desc_Centralization, r_desc_PEOE9, r_qp_CIQPlogS, i_desc_Ring_Count_5, i_desc_Sum_of_topological_distances_between_N..N, r_qp_QPlogHERG, r_qp_QPlogKhsa, i_desc_Number_of_ring_systems, r_desc_First_Mohar, r_qp_QPlogS, r_desc_Global_topological_charge, r_desc_Narumi_Harmonic_Topological, r_qp_glob, r_desc_Mean_topological_charge_index_of_order_2, i_qp_#in56

We selected the most valuable features from the set and reduced the number of features from 243 to 42, selecting the first 6 features from the Chi-Square method, the first 6 features from the Info Gain method, the first 5 features from the Gini Index method, the first 15 features from the Pearson Correlation Coefficient method and the first 10 features from the Fisher Ratio method for a total of 42, achieving a reduction to the initial features by 82.7%. With the extracted features we trained our models again scoring almost identical accuracy scores with the initial models and reducing the train time from 1′5" to 12" achieving 81.5% time reduction. The results can be seen in the table below:

Model	Train Accuracy	**Test Accuracy**	Completion Time
Random Forest	61.38%	62.87%	2.234 sec
Gradient Boosting	60.70%	60.39%	8.304 sec
MLP	60.94%	58.91%	5.411 sec
Nearest Neighbors	63.92%	57.92%	0.042 sec
Logistic Regression	60.10%	57.92%	0.345 sec
SVM (RBF)	58.38%	55.94%	0.148 sec
AdaBoost	55.40%	53.46%	1.153 sec
Naive Bayes	53.69%	53.46%	0.017 sec
Decision Tree	56.75%	50.90%	0.100 sec
QDA	47.58%	44.55%	0.048 sec

After we studied the first data set, we proceeded into testing the second data set. Firstly, we tried to train our models, with the same principal as the first one, so that they can classify the data for the classes 1 and 2. Class 1 being a compound with anti-inflammatory activity and class 2 being a compound with antioxidant activity. The results are shown in the table below:

Model	Train Accuracy	**Test Accuracy**	Completion Time
Random Forest	98.61%	98.76%	16.663 sec
Gradient Boosting	97.96%	98.55%	109.003 sec
Decision Tree	97.67%	98.07%	4.987 sec
AdaBoost	97.85%	97.45%	23.145 sec
MLP	94.96%	95.81%	66.063 sec
QDA	95.22%	94.98%	1.993 sec
Nearest Neighbors	94.14%	93.81%	2.374 sec
Logistic Regression	92.81%	92.58%	1.106 sec
SVM (RBF)	90.57%	90.24%	6.405 sec
Naive Bayes	77.70%	78.77%	0.188 sec

Then, we tried to train our models to be able to classify data for the classes 1 and 3. Class 3 being a anti-inflammatory decoy compound. The results are shown in the table below:

Model	Train Accuracy	**Test Accuracy**	Completion Time
Random Forest	95.63%	96.37%	31.350 sec
Decision Tree	93.32%	94.29%	9.382 sec
Gradient Boosting	91.11%	91.03%	201.028 sec
AdaBoost	86.38%	85.74%	46.243 sec
QDA	79.08%	82.52%	2.859 sec
Nearest Neighbors	81.68%	82.28%	4.812 sec
Naive Bayes	73.54%	72.70%	0.333 sec
MLP	72.13%	71.56%	122.106 sec
Logistic Regression	70.34%	68.47%	2.133 sec
SVM (RBF)	64.04%	63.29%	92.964 sec

Then, we tried to train our models to be able to classify data for the classes 2 and 4. Class 4 being a antioxidant decoy compound. The results are shown in the table below:

Model	Train Accuracy	**Test Accuracy**	Completion Time
Random Forest	89.25%	90.00%	5.366 sec
Gradient Boosting	89.79%	89.79%	35.161 sec
AdaBoost	84.69%	87.50%	8.482 sec
Decision Tree	84.50%	84.16%	1.077 sec
MLP	77.25%	83.33%	25.269 sec
QDA	76.09%	80.20%	1.460 sec
Nearest Neighbors	76.45%	78.12%	0.693 sec
Logistic Regression	74.48%	75.83%	0.595 sec
SVM (RBF)	69.47%	72.50%	2.348 sec
Naive Bayes	54.96%	55.62%	0.059 sec

After that, we tried to train our models to be able to classify data among three classes. Firstly, we tried to train our models to be able to classify data for the classes 1, 2 and 3. The results are shown in the table below:

Model	Train Accuracy	**Test Accuracy**	Completion Time
Random Forest	94.89%	95.76%	35.148 sec
Decision Tree	92.12%	92.87%	10.149 sec
Gradient Boosting	90.63%	90.60%	665.753 sec
QDA	79.40%	79.27%	3.384 sec
Nearest Neighbors	78.95%	78.58%	5.277 sec
MLP	63.88%	73.61%	27.483 sec
AdaBoost	72.22%	72.99%	50.366 sec
Logistic Regression	64.50%	62.86%	5.396 sec
Naive Bayes	62.57%	60.01%	0.368 sec
SVM (RBF)	60.92%	59.46%	112.920 sec

Then, we tried to train our models to be able to classify data for the classes 1, 2 and 4. The results are shown in the table below:

Model	Train Accuracy	**Test Accuracy**	Completion Time
Random Forest	94.91%	95.48%	27.784 sec
Gradient Boosting	93.74%	94.53%	420.551 sec
QDA	90.63%	92.86%	2.958 sec
Decision Tree	92.18%	92.69%	6.698 sec
Nearest Neighbors	82.56%	83.94%	3.198 sec
AdaBoost	83.85%	82.99%	32.896 sec
MLP	76.06%	80.60%	21.493 sec
Logistic Regression	75.32%	77.98%	3.507 sec
SVM (RBF)	73.10%	74.86%	29.420 sec
Naive Bayes	58.86%	57.75%	0.238 sec

After that, we tried to train our models to be able to classify data among all the classes of the data set, that being 1, 2, 3 and 4. The results are shown in the table below:

Model	Train Accuracy	**Test Accuracy**	Completion Time
Random Forest	93.22%	93.96%	41.474 sec
Decision Tree	89.58%	89.77%	10.647 sec
Gradient Boosting	88.35%	89.40%	1042.295 sec
QDA	76.68%	77.70%	4.106 sec
Nearest Neighbors	73.03%	75.39%	5.858 sec
AdaBoost	67.94%	70.72%	58.016 sec
MLP	61.24%	65.09%	29.609 sec
Logistic Regression	56.43%	54.26%	5.357 sec
SVM (RBF)	53.95%	52.52%	157.473 sec
Naive Bayes	52.30%	52.01%	0.404 sec

Concerning the second data set we trained effectively 10 models with all possible combinations of classes and achieved accuracy scores up to $98,8\%$ with the top 5 models being : Random Forest, Decision Tree, Gradient Boosting, QDA and AdaBoost scoring on average 95%, 92%, 91.9%, 84.5% and 82.8% respectively. Now we are going to give as an input the 39, lab tested, compounds to our models to see the categorical predictions and compare them to the lab results. The results can be seen in the table below:

Compound ID	AdaBoost	Decision Tree	Random Forest	Gradient Boosting	QDA	Lab Results
1	1	1	1	1	1	1
1	1	1	1	1	1	1
3	1	1	1	1	1	1
3	1	1	1	1	1	1
3	1	1	1	1	1	1
3	1	1	1	1	1	1
4	1	1	1	1	1	1
4	1	1	1	1	1	1
5	1	1	1	1	1	1
5	1	1	1	1	1	1
6	1	1	1	1	1	1
6	1	1	1	1	1	1
6	1	1	1	1	1	1
6	1	1	1	1	1	1
9	1	1	1	1	1	1
9	1	1	1	1	1	1
10	1	2	1	1	1	1

Compound ID	AdaBoost	Decision Tree	Random Forest	Gradient Boosting	QDA	Lab Results
11	1	1	1	1	1	1
12	1	1	1	1	1	1
13	1	1	1	1	1	1
14	1	1	1	1	1	1
15	1	1	1	1	1	1
16	1	1	1	1	1	1
17	1	1	1	1	1	1
17	1	1	1	1	1	1
18	2	1	1	1	1	2
18	2	1	1	1	1	2
19	2	1	1	1	1	2
19	2	1	1	1	1	2
20	2	2	2	2	1	2
22	2	2	1	2	1	2
23	2	2	1	1	1	2
25	2	1	1	1	1	2
25	2	1	1	1	1	2
26	1	2	1	1	1	2
26	1	2	1	1	1	2
27	2	2	1	2	1	2
28	1	1	2	2	1	2
29	2	2	1	1	1	2

After testing the 39 compounds on our top 5 models, we achieved an extremely good accuracy of 92.3% using the AdaBoost model predicting correctly 36 out of 39 compounds in 1′15”. Additionally we selected the most valuable features from the data set for future studies, using the same methods to the first data set. The results, which are organized as before, can be seen in the table below:

Evaluator	Feature Ranking
Chi-Square	i_desc_Sum_of_topological_distances_between_P..P, i_qp_#amide, i_qp_#acid, i_desc_Sum_of_topological_distances_between_O..P, r_desc_Total_structure_connectivity, r_desc_PEOE8, i_desc_Sum_of_topological_distances_between_N..P, r_qp_SAamideO, i_desc_Sum_of_topological_distances_between_O..O, i_desc_Ring_Count_13, i_desc_Sum_of_topological_distances_between_P..F, i_desc_Sum_of_topological_distances_between_S..P, r_desc_PEOE6, r_desc_Average_valence_connectivity_index_chi-2, i_qp_#in34
Info Gain	r_desc_Balaban-type_index_from_electronegativity_weighted_distance_matrix, r_desc_MR8, r_desc_Van_der_Waals_surface_area, r_desc_Kier_benzene, r_desc_Wiener-type_index_from_Z_weighted_distance_matrix_-_Barysz_matrix, r_desc_Wiener-type_index_from_polarizability_weighted_distance_matrix, r_desc_Balaban-type_index_from_polarizability_weighted_distance_matrix, r_desc_Average_valence_connectivity_index_chi-1, r_desc_Wiener-type_index_from_van_der_waals_weighted_distance_matrix, r_desc_Average_valence_connectivity_index_chi-2, r_desc_Average_valence_connectivity_index_chi-3, r_desc_Average_valence_connectivity_index_chi-4, r_desc_Balaban-type_index_from_Z_weighted_distance_matrix_-_Barysz_matrix, r_desc_Balaban-type_index_from_mass_weighted_distance_matrix
Gini Index	i_desc_Ring_Count_13, i_desc_Ring_Count_12, i_desc_Ring_Count_11, i_desc_Ring_Count_10, i_desc_Gutman_Molecular_Topological, i_desc_Centralization, i_desc_Number_of_ring_systems, i_desc_Eccentric_connectivity, i_desc_Eccentricity, i_desc_Cyclomatic_number, i_desc_Quadratic, i_desc_Atoms_in_Ring_System, i_desc_Bonds_in_Ring_System, i_desc_First_Zagreb, i_desc_Bond_Count
Pearson Correlation Coefficient	r_qp_QPlogS, r_desc_PEOE1, i_desc_Sum_of_topological_distances_between_O..O, r_desc_Total_structure_connectivity, r_qp_CIQPlogS, r_desc_Mean_topological_charge_index_of_order_4, r_desc_ALOGP6, r_qp_FISA, r_desc_Balaban-type_index_from_mass_weighted, r_desc_Balaban-type_index_from_Z_weighted, r_qp_ACxDN, r_desc_Balaban-type_index_from_electronegativity, r_desc_Balaban_distance_connectivity_index, r_desc_PEOE12, r_desc_Mean_topological_charge_index_of_order_3
Fisher Ratio	r_desc_Total_structure_connectivity, r_desc_Radial_centric, r_desc_PEOE1, r_qp_PercentHumanOralAbsorption, r_desc_Eccentric, r_desc_Average_eccentricity, r_desc_Average_valence_connectivity_index_chi-0, r_desc_Second_Mohar, r_desc_Mean_Square_Distance_Balaban, i_desc_Topological_diameter, r_qp_QPlogKp, r_desc_Mean_Wiener, i_desc_Topological_radius, r_desc_Average_valence_connectivity_index_chi-1, r_desc_Kier_benzene-likeliness_index

Finally we studied our third data set by training 5 Regression models, using 70% of our data set and the rest 30% for training. We manged to achieve an accuracy of 85.1% at max, using the Random Forest Regressor model. The results are shown in the table below:

Model	Train Accuracy	**Test Accuracy**	Completion Time
Random Forest Regressor	84.20%	85.16%	470.188 sec
Decision Tree Reggresor	69.78%	72.23%	7.669 sec
Gradient Boosting Regressor	71.22%	70.49%	4.106 sec
Linear Regressor	61.70%	62.08%	1.188 sec
Support Vector Regressor	22.77%	24.94%	82.205 sec

Then we tested the 24, lab tested, compounds using the Random Forest Regressor model achieving a 75.2% accuracy in 0.009". The results can be seen in the table below:

Compound ID	Random Forest Regressor	Lab Results
1	5.7696	6
3	6.49066667	5.52
4	6.33026667	5.4
5	6.40666667	5.3
6	6.2983	5.22
9	6.5294	5.05
10	6.2304	5
11	6.225	4.69
12	5.8905	4.92
13	6.3804	4.98

Compound ID	Random Forest Regressor	Lab Results
14	6.1345	4.85
15	6.3059	4.82
16	6.1893	4.8
17	5.9132	4.77
18	5.1568	4.74
19	5.8331	4.72
20	5.89550115	4.7
22	5.90148334	4.66
23	5.8925	4.64
25	5.5213	4.6
26	5.9308	4.59
27	6.31160115	4.57
28	5.9453	4.55
29	6.0762	4.55

Additionally we extracted the most valuable features from the data set, using the methods that have been used in the two previous data sets. The features can be used for future studies. The results can be seen in the table below:

Evaluator	Feature Ranking
Chi-Square	Ring Count 4 Chirality count, Ring Count 3, Ring Count 7, Topological charge index of order 8, Topological charge index of order 7, Ring Count 5, Topological charge index of order 9, Topological charge index of order 6, Topological charge index of order 10, Topological charge index of order 5, Molecular electrotopological variation, Topological charge index of order 3, Total structure connectivity, E-state topological parameter
Info Gain	NDDO Heat of Formation, glob, volume,SASA, QPPCaco, PISA, LUMO Energy, HOMO Energy, FISA, PSA, Wiener-type index from electronegativity weighted, Wiener-type index from mass weighted distance, Wiener-type index from Z weighted distance matrix, Wiener-type index from polarizability weighted, Wiener-type index from van der waals weighted distance
Gini Index	Total structure connectivity, Ring Count 4, #amidine, #rtvFG, #amide, Chirality count, PEOE12, PEOE14, CNS, PEOE1, Average valence connectivity index chi-5, Average connectivity index chi-1, Average valence connectivity index chi-4, PEOE4, PEOE2
Pearson Correlation Coefficient	Number of ring systems, ALOGP4, Sum of topological distances between N..N, Atoms in Ring System, #ringatoms, Ring perimeter, ALOGP7, Variation, Bonds in Ring System, Topological diameter, Radial centric, #in56, reciprocal distance Randic-type index, PEOE3, Topological radius
Fisher Ratio	Sum of topological distances between S..Br, Sum of topological distances between F..Br, Mean topological charge index of order 3, Average valence connectivity index chi-4, Sum of topological distances between O..Cl, Sum of topological distances between O..Br, Topological charge index of order 8, Topological charge index of order 6, Topological charge index of order 7, Sum of topological distances between N..Br, Chirality count, Topological charge index of order 9, Topological charge index of order 3, Topological charge index of order 5, Topological charge index of order 10

Chapter 5

Conclusions

In this study we have presented how machine learning and computer science can upgrade the health care community. Firstly, we explained the concept of machine learning engineering , its main principles and then the aspects of supervised learning problems. Then we explained the nature of the problem we set to solve and analyzed in detail the properties of the enzyme lipoxygenase (LOX). Given the pharmaceutical data, we tried to categorize the different types of compounds using a wide variety of machine learning algorithms as well as predict the exact number of effectiveness of specific proteins. We managed to categorize correctly the compounds given the ground truth in some cases and in others not, with accuracy scores up to 98.8%. Furthermore we gave mathematical meaning to each molecular descriptor of the data sets so we could single out and pick the best features that had the highest correlation among the others. From the analysis of the data we can conclude that machine learning has the potential of helping the medical society with pharmaceutical experiments. Possible future improvements can be made using the extracted features as well as the trained models for use on other enzymes and proteins.

List of Figures

Bibliography

[1] Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. Cambridge University Press, 2009.

[2] Per-Erik Danielsson. Euclidean distance mapping. *Computer Graphics and image processing*, 14(3):227–248, 1980.

[3] Tran Huy Dat and Cuntai Guan. Feature selection based on fisher ratio and mutual information analyses for robust brain computer interface. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 1, pages I–337. IEEE, 2007.

[4] Stephan Dreiseitl and Lucila Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics*, 35(5-6):352–359, 2002.

[5] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[6] Benyamin Ghojogh and Mark Crowley. Linear and quadratic discriminant analysis: Tutorial. *arXiv preprint arXiv:1906.02590*, 2019.

[7] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

[8] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[9] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural Networks for Perception*, pages 65–93. Elsevier, 1992.

[10] Julien IE Hoffman. *Biostatistics for medical and biomedical practitioners*. Academic Press, 2015.

[11] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[12] James M Keller, Michael R Gray, and James A Givens. A fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, (4):580–585, 1985.

[13] Aleix M Martinez and Avinash C Kak. Pca versus lda. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):228–233, 2001.

[14] S Martınez, F Nicolás, F Pennini, and A Plastino. Tsallis' entropy maximization procedure revisited. *Physica A: Statistical Mechanics and its Applications*, 286(3-4):489–502, 2000.

[15] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[16] Mahesh Pal. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222, 2005.

[17] Heum Park and Hyuk-Chul Kwon. Improved gini-index algorithm to correct feature-selection bias in text classification. *IEICE Transactions on Information and Systems*, 94(4):855–865, 2011.

[18] Harsh H Patel and Purvi Prajapati. Study and analysis of decision tree based classification algorithms. *International Journal of Computer Sciences and Engineering*, 6(10):74–78, 2018.

[19] S. Gopal Krishna Patro and Kishore Kumar Sahu. Normalization: A preprocessing stage. *CoRR*, abs/1503.06462, 2015.

[20] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[21] Payam Refaeilzadeh, Lei Tang, and Huan Liu. *Cross-Validation*, pages 532–538. Springer US, Boston, MA, 2009.

[22] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, volume 3, pages 41–46, 2001.

[23] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

[24] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[25] Rania Saidi, Waad Bouaguel, and Nadia Essoussi. Hybrid feature selection method based on the genetic algorithm and pearson correlation coefficient. In *Machine Learning Paradigms: Theory and Application*, pages 3–24. Springer, 2019.

[26] Albert Satorra and Peter M Bentler. A scaled difference chi-square test statistic for moment structure analysis. *Psychometrika*, 66(4):507–514, 2001.

[27] Wenqian Shang, Houkuan Huang, Haibin Zhu, Yongmin Lin, Youli Qu, and Zhihai Wang. A novel feature selection algorithm for text categorization. *Expert Systems with Applications*, 33(1):1–5, 2007.

[28] Vijaykumar Sutariya, Anastasia Groshev, Prabodh Sadana, Deepak Bhatia, and Yashwant Pathak. Artificial neural network in drug delivery and pharmaceutical research. *The Open Bioinformatics Journal*, 7(1), 2013.

[29] Shan Suthaharan. Support vector machine. In *Machine Learning Models and Algorithms for Big Data Classification*, pages 207–235. Springer, 2016.

[30] Evanthia Tripoliti and George Manis. Feature selection in hrv analysis of young and elderly subjects. In *5th European Conference of the International Federation for Medical and Biological Engineering*, pages 516–519. Springer, 2011.

[31] Ariadni Tzara, George Lambrinidis, and Angeliki Kourounakis. Design of multifaceted antioxidants: shifting towards anti-inflammatory and antihyperlipidemic activity. *Molecules*, 26(16):4928, 2021.