UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

GABRIEL MARANGONI MOITA

Combining Performance and Diversity Measures for Optimizing Classification Ensembles via a Genetic Algorithm in the miRNA-Target Prediction Problem

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Science

Advisor: Profa. Dra. Mariana R. Mendoza

Porto Alegre
December 2018

"The Answer to the Great Question... Of Life, the Universe and Everything... Is... Forty-two."

— Deep Thought, with infinite majesty and calm.

ACKNOWLEDGMENTS

I would like to thank my advisor, Prof. Dr. Mariana Recamonde Mendoza, for her unstoppable (and needed) feedbacks and contributions during this year of research, pursuing always the best for our work, even with all difficulties. I wish the best for you and your son in the years to come. I'm also grateful for all classes that I received during the course, with a special mention to Prof. Dr. Bruno Castro da Silva and again Prof. Dr. Mariana Recamonde Mendoza for the deep knowledge in Artificial Intelligence that I've acquired during their classes. Thanks to all professors that I've interacted during my journey through this course; the complete professional I'm today is due to the things I've learned with their help.

I would also want to thank the support of my family since I was born. If I'm writing this monography today, is due to their perseverance in always seeking for the best education for me, and also supporting me in all moments. Thanks father, Erico Manoel Alves Moita, and mother, Alessandra Jardim Marangoni Moita, for everything. Also thanks to my younger brother Nícolas Marangoni Moita for all the expended hours in our shared computer that prevented him from playing as much as he would like, and my younger sisters Nina Luísa Marangoni Moita and Marianna Vitória Marangoni Moita for (almost) always respecting my requests for silence when I needed, and to all my siblings for bringing a smile to my face when I needed one. And a final thanks to Mikaela Silva de Oliveira, for warming my heart and for all the help through all the turbulence we've been facing together.

ABSTRACT

MicroRNAs, also called miRNAs, are a large family of non-coding RNAs of approximately 22 nucleotides (nt) in length, which act as post-transcriptional gene silencers via translational repression or degradation of targets mRNAs, and have an important role in metabolism and genesis of different genetic diseases, such as cancers. The miRNA target prediction problem is considered a difficult challenge in the molecular biology area. There are millions of possible miRNA-mRNA possible combinations, and to experimentally find the functional combinations takes a large quantity of effort, therefore time and investment.

The scientific community is actively researching computational approaches to overcome this cost with Machine Learning and their predictive models to better understand the interactions between miRNA-mRNA, and how they influence metabolic and disease processes. The purpose of this work is to study the effect of combining performance and diversity measures in a Genetic Algorithm's (GA) fitness function that learns the best combination of classifiers in an heterogeneous ensemble classifier in the miRNA-Target prediction problem.

Through experimentation, we've concluded that the challenge presented by the unbalanced and relatively small available datasets overshadows the possible benefits that the diversity measure could bring to the GA fitness function. Although the ensemble optimization combining performance and diversity measures has achieved better solutions than performance-based optimization in some cases, on average, the former solution did not surpass the latter. This doesn't allow us to conclude if the combination of performance and diversity measures results in better ensembles or not in our problem.

Keywords: MicroRNA Target Prediction. Ensemble Learning. Genetic Algorithms. Diversity Measures.

Combinando Medidas de Performance e Diversidade para Otimizar Classificadores Ensemble através de Algoritmo Genético no Problema de Predição de Alvos de miRNAs

RESUMO

MicroRNAs, também chamados de miRNAs, são uma grande família de RNAs não-codificantes de aproximadamente 22 nucleotídeos (nt) de tamanho, que atuam como silenciadores pós-transcricionais de genes através da repressão da tradução ou degradação dos mRNAs alvos, e tem um papel importante no metabolismo e na criação de diferentes doenças genéticas, como cânceres. O problema da predição de alvos de miRNAs é considerado um difícil desafio na área de biologia molecular. Há milhões de possíveis combinações entre miRNAs e mRNAs, e encontrar experimentalmente as combinações funcionais demanda um grande esforço, ou seja, tempo e investimento.

A comunidade científica está ativamente pesquisando abordagens computacionais para superar esses custos com *Machine Learning* e seus modelos preditivos para melhor entender a interação entre miRNAs e mRNAs, e como eles influenciam nos processos metabólicos e de doenças. O propósito deste trabalho é estudar o efeito da combinação de medidas de performance e diversidade na função de *fitness* de um Algoritmo Genético (AG) que aprende a melhor combinação de classificadores em um classificador conjunto heterogêneo no problema da predição de alvos de miRNAs.

Através de experimentação, nós concluímos que o desafio apresentado pelos *datsets* desbalanceados e relativamente pequenos obscurece os possíveis benefícios que a medida de diversidade pode trazer para a função de *fitness* do AG. Embora a otimização do ensemble combinando medidas de performance e diversidade tenha alcançado soluções melhores do que optimização baseada em performance em alguns casos, na média ela não supera. Isso não nos permite concluir se a combinação das medidas de performance e diversidade resulta ou não em conjuntos melhores no nosso problema.

Palavras-chave: Predição de Alvos de microRNAs, Aprendizado Ensemble, Algoritmos Genéticos, Métricas de diversidade.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

RNA	Ribonucleic acid
mRNA	Messenger RNA
miRNA	Micro RNA
AI	Artificial Intelligence
ML	Machine Learning
GA	Genetic Algorithm

CONTENTS

1 INTRODUCTION

MicroRNAs, also called miRNAs, are a large family of non-coding RNAs of approximately 22 nucleotides (nt) in length, which act as post-transcriptional gene silencers via translational repression or degradation of targets mRNAs (FILIPOWICZ; BHATTACHARYYA; SONENBERG, 2008; YUE; LIU; HUANG, 2009; BARTEL, 2004).

The microRNA (also called miRNA) target prediction problem is considered a difficult challenge in the molecular biology area. There are millions of possible miRNA-mRNA combinations, the majority of them being non-functional. These interactions are important due to its regulatory role in genetic expression, influencing metabolic processes that imply many diseases, such as tumor genesis and several types of cancer (HE; HANNON, 2004; YANAIHARA *et al.*, 2006; KLUIVER *et al.*, 2005), and therefore need to be studied.

Currently, the available datasets have thousands of experimentally verified examples and very few non-functional ones (YU *et al.*, 2014), thus generating a great unbalance towards the functional samples. The scientific community is actively researching computational approaches to overcome the cost of manual experiments to determine whether a given miRNA-mRNA pair is functional. Following this direction, Machine Learning (ML) algorithms were proven promising to train predictive models and better understand the miRNA-mRNA interactions and its influence in the metabolism. Among these computation approaches, ensemble learning variations are commonly used due to its ability to have a good performance even with challenges such as small and unbalanced datasets, as observed in the current domain.

Our proposal is to use an heterogeneous ensemble learning to aggregate different classifiers into one single, global decision, as already seen in related work. To choose which combination of classifiers is the best, we use a Genetic Algorithm where the usage or not of each classifier is encoded in the individuals' genotypes, and optimized according to different criteria. In particular, in the scope of this work, our goal is to investigate if the inclusion of a diversity measure together with a performance measure in the Genetic Algorithm fitness function, combined with different proportions, will improve the overall performance of the algorithm, *i.e.*, will make it build better ensembles, when compared to solutions using only the performance measure in the fitness function.

This work is organized as follows: Chapter 2 presents the required theoretical background to understand our proposed technique; Chapter 3 presents related work; Chapter 4 presents our proposal for this problem; Chapter 5 presents the experimental results, with comparisons between different diversity proportions and against the individual classifiers and the possible full ensemble; and Chapter 6 presents the conclusion.

2 THEORETICAL BACKGROUND

2.1 Biological Background

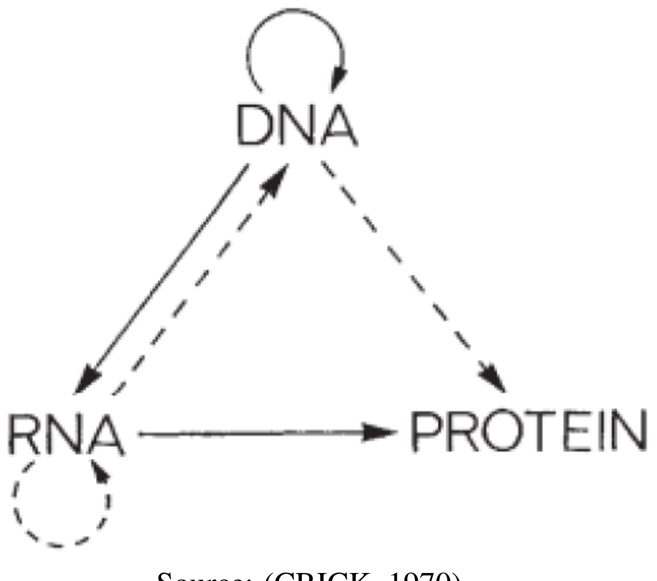
2.1.1 Central Dogma of Molecular Biology

In Biology, the field of Molecular Biology is responsible for studying cellular molecules, such as nucleic acids and proteins, including their composition, structure and interactions. These elements are relevant to the cell's functionality and maintenance, as they are the essential biological processes inside them. The next paragraphs are based on Zaha, Ferreira and Passaglia (2014).

The DNA (**D**eoxyribo**n**ucleic **A**cid) is composed of two chains of nucleotides in a double helix pattern. The nucleotides encode all genetic instructions used in many functions of any living being, such as reproduction, growth and development. To convert all these genetic instructions in actions, they need to be translated into proteins that will influence the cells behavior, in a process called **genetic expression**.

Crick (1970) determines the famous Central Dogma of Molecular Biology, explaining the information flow from DNA to RNA and proteins. A classification is shown in Figure 2.1 below.

Figure 2.1: Molecular biology information flow. Solid arrows show general transfers; dotted arrows show special transfers. The absent arrows are the undetected transfers specified by the central dogma.

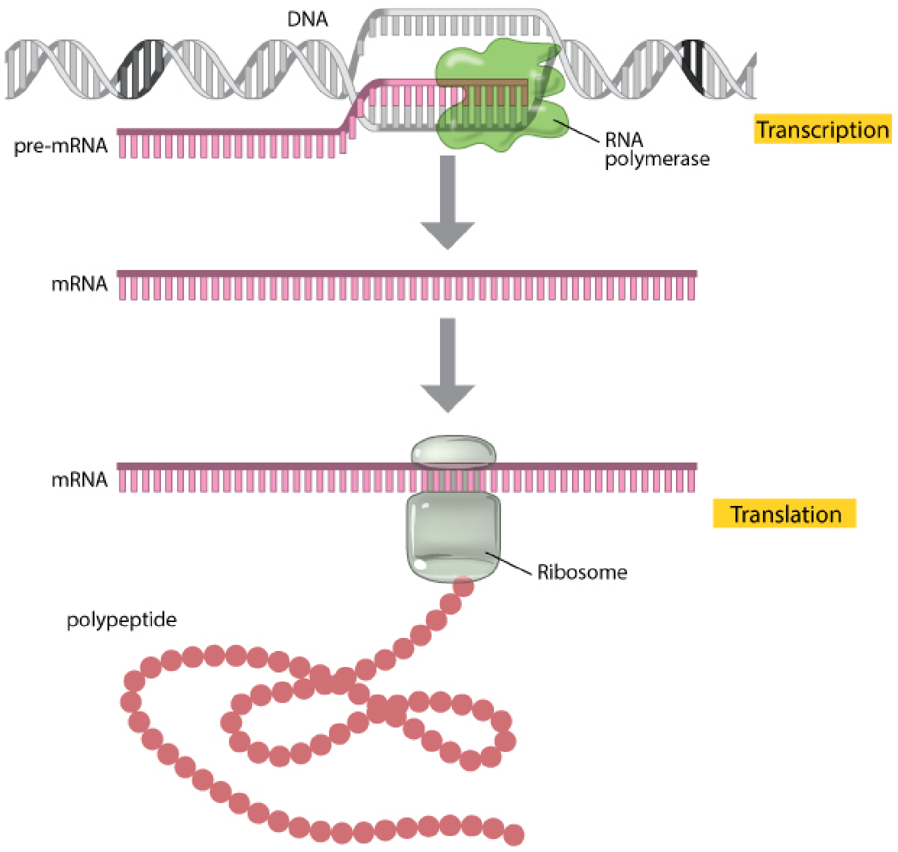


Source: (CRICK, 1970).

The process of converting an DNA gene into a RNA (**R**ibo**n**ucleic **A**cid) is called

transcription. In this process, the DNA double helix is used as an template by an enzyme called **RNA polymerase** to synthesise a single chain of nucleotides, called **mRNA** (**Messenger** RNA). This mRNA is then carried to the ribosome, where it is **translated** into a amino acid chain (or **polypeptide**), which is later folded into an active protein. This whole process is exhibited in Figure 2.2.

Figure 2.2: Genetic expression process.

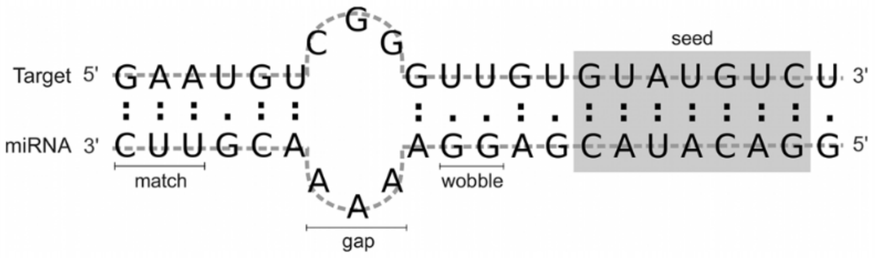


Source: (CLANCY; BROWN, 2008).

2.1.2 MicroRNA and their interaction with mRNA

MicroRNAs, also called miRNAs, are a large family of non-coding RNAs of approximately 22 nucleotides (nt) in length, which act as post-transcriptional gene silencers via translational repression or degradation of targets mRNAs (FILIPOWICZ; BHATTACHARYYA; SONENBERG, 2008; YUE; LIU; HUANG, 2009; BARTEL, 2004). In other words, the miRNAs silences target mRNAs repressing their translation into polypeptides, interrupting the genetic expression process. One example of miRNA-mRNA alignment is given in Figure 2.3.

Figure 2.3: Example of miRNA-target alignment. Nucleotides matches are shown by colons and G:U wobble pairs are represented by dots. There can be gaps.


Target 5' G A A U G U C G G G U U G U G U A U G U C U 3'
miRNA 3' C U U G C A A A A A G G A G C A U A C A G G 5'
seed
match
wobble
gap

Source: (MENDOZA *et al.*, 2013).

The study of microRNA-mRNA interactions is important due to its role in regulating genetic expression in metabolic processes, such as developmental timing, growth, cell proliferation and defense against viruses (LU *et al.*, 2008). This way, miRNAs play a regulatory role in many biological processes and diseases, including tumor genesis of several types of cancer (HE; HANNON, 2004; YANAIHARA *et al.*, 2006; KLUIVER *et al.*, 2005).

There are millions of possible miRNA-mRNA combinations, as one miRNA can be regulate many targets, and one mRNA can be targeted and regulated by many miRNA. There isn't a deep knowledge of the basic mechanisms related with target recognition yet (STURM *et al.*, 2010). Identification of miRNAs' targets is really important to better understand the biological mechanism and find new ways to combat cancer and other genetic-related diseases.

There are many databases containing miRNA-gene information, but they provide manually curated experimentally validated interactions (VLACHOS *et al.*, 2015), leading to a low number of human entries compared with the exponentially high number of possible combinations of nucleotides. Experimentally determine the miRNA-RNA interactions is expensive, however, Machine Learning can help with its predictive models to gather more knowledge in a cheaper and faster way.

2.2 Computational Background

2.2.1 Genetic Algorithms

Genetic Algorithms (GAs) are a family of meta-heuristics that emulate the natural selection process described by Darwin (1859), using concepts such as fitness selection, genetic mutation and crossover. It's a subclass of Evolutionary Algorithms.

The conception of this type of algorithm trace back to Alan Turing in 1950, when he proposes in Turing (1950) a learning machine which would parallelize the principles of evolution. Afterwards, beginning in 1957, a series of publications by Alex Fraser report the simulation of artificial selection of individuals, including all modern algorithms' essential concepts, making usage of the Monte Carlo method (FRASER, 1957). More recently, NASA used a GA to fully develop antennas with complex shapes for difficult objectives, for example, the one used in the Space Technology 5 mission (HORNBY *et al.*, 2006).

A Genetic Algorithm comprises a loop of different phases of execution, until the solution is considered to have been found. The different phases are:

- Initial Population Generation
- Fitness Evaluation
- Selection
- Recombination (or Crossover)
- Mutation
- Termination Criteria Evaluation

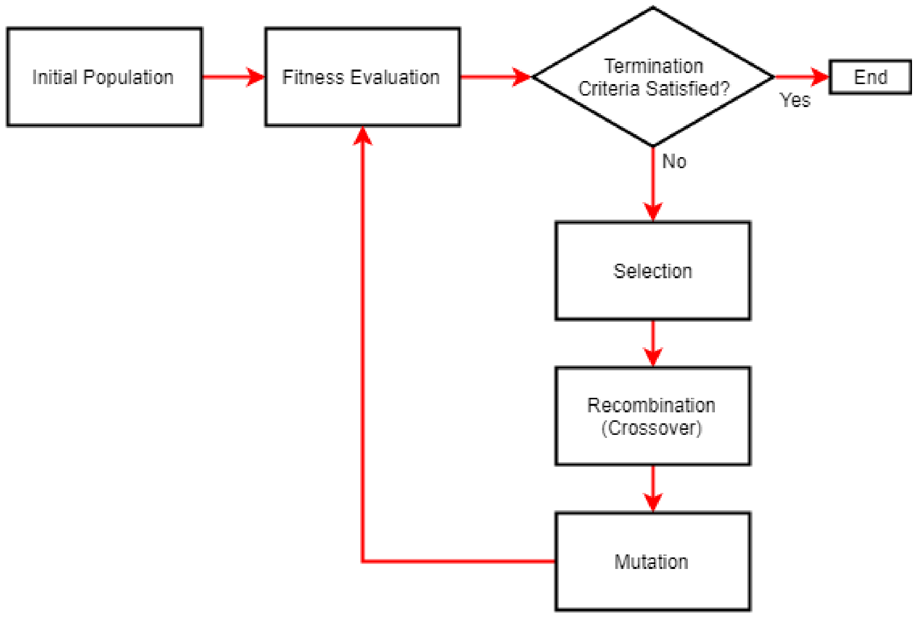
The loop itself and how the forementioned phases follow each other are illustrated in the following figure:

In the next sections, each phase will be explained in detail, demonstrating different algorithms that can be used in each one.

2.2.1.1 Individual Chromosome Representation

Individuals in a GA are represented by their chromosomes. The semantic of a chromosome reflects the nature of the problem, two different chromosomes needs to refer to two different solutions that will be evaluated through the GA process. A binary rep-

Figure 2.4: Genetic Algorithm Loop.



Source: Elaborated by the Author.

resentation might represent some kind of Boolean logic, with $0 = false$ and $1 = true$ in each position of the chromosome meaning, for example, if the item i is used or not in the solution. A non-binary representation might represent labels, *e.g.* nodes in a graph and the order meaning the order they're visited, or quantities, *e.g.* how much of the item i used in the solution.

In this work, we'll be using a binary representation. It will be further explained in the Proposal (Chapter 4).

2.2.1.2 Initial Population Generation

To start the learning process in a Genetic Algorithm, we need an initial population to work with. It has a large impact in the performance of the algorithm, as it defines which areas of the search space will be first explored.

It can be created due to some specific heuristic related to the problem domain with previous knowledge, or with a random process. In a binary representation, the most common way to generate the first individuals is with the latter, generating each position of the chromosome with equal chances of being 0 and 1, *i.e.*, an **uniform distribution**). This is the method used in this work.

2.2.1.3 Fitness Functions

The fitness function defines how good an individual is, *i.e.*, how close to the optimal solution the solution is. It needs to represent it quantitatively, in a way that a worse solution has a worse score value than a good one. Also, it needs to be easy to calculate, as it will be evaluated for every individual in every generation, being a common bottleneck in GAs.

In this work, the fitness will be a performance measure, combined (or not) with a diversity measure. Further explanation will be given in the Proposal (Chapter 4).

2.2.1.4 Selection Methods

The selection method defines how individuals from the population will be chosen to compose the new generation. It should follow the evolution theory, where the fittest individuals shall have more chances of reproducing. The methods are executed k times to select the k individuals required by the Crossover Method.

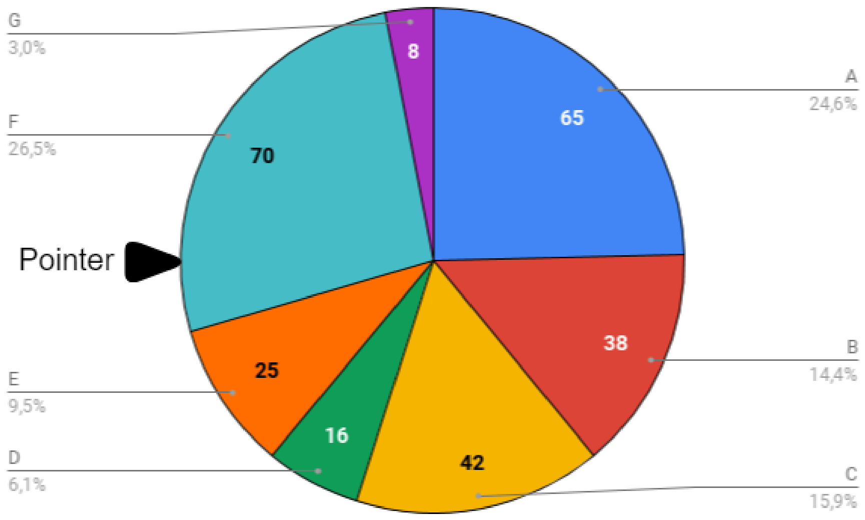
Below are two examples of selection methods:

- Roulette-Wheel Selection
  Also called *fitness proportionate selection*, this method gives every individual a chance of being selected proportionally to its fitness. A possible visualization is a roulette-wheel in a casino, where the size of each individual is equivalent to its score value, and the selection is equivalent to spinning the wheel to see which one the pointer will land on.
  Mathematically, if f_i is the fitness of the individual i and N is the number of individuals in the current population, the probability p_i of the individual i being chosen is defined as:

$$p_i = \frac{f_i}{\Sigma_{j=1}^{N} f_j} \tag{2.1}$$

- Tournament Selection
  This method consists into randomly choosing a pre-set number of individuals in the population, and then select the fittest of them with probability p, the second fittest with probability $p \cdot (1 - p)$, the third fittest with probability $p \cdot ((1 - p)^2)$, and so on. When $p = 1$, it is called a *deterministic* tournament selection. It is most commonly used in practice, as it suffers less from stochastic noise, and has an

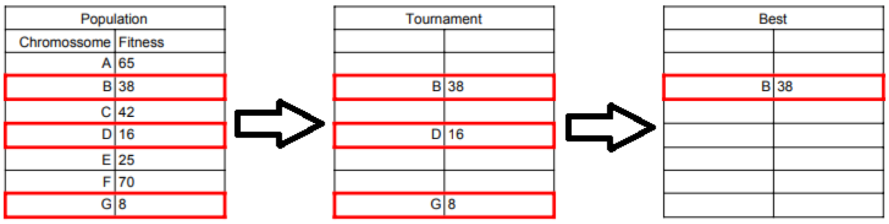
Figure 2.5: Roulette-Wheel Selection example.



Source: Elaborated by the Author.

adjustable selection pressure. This is the method used in this work.

Figure 2.6: Tournament Selection example, with $size = 3$ and $p = 1$.



Population	
Chromossome	Fitness
A	65
B	38
C	42
D	16
E	25
F	70
G	8

Tournament	
B	38
D	16
G	8

Best	
B	38

Source: Elaborated by the Author.

2.2.1.5 Crossover Methods

Also called *recombination* methods, the crossover methods defines how the selected individuals are combined. In other words, they are different genetic operators used to combine the genetic information from the two individuals to generate their offspring. It is equivalent to the genetic crossover that happens in biological sexual reproduction.

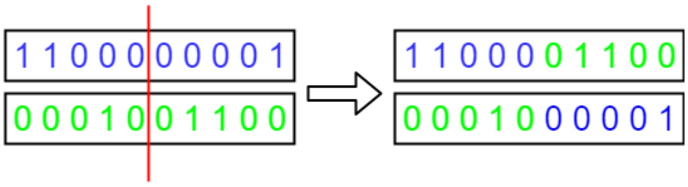
Crossover methods are commonly activated considering a *crossover rate*. A number between 0 and 1 is chosen for each group of selected parents. If it is lower or equal to the crossover rate, the method is applied, otherwise the parents doesn't breed and go to the next generation. The crossover rate value should be above 0.5 to make sure enough combinations of solutions are performed in order to the GA to work, as a value equal to zero would lead to no genetic recombination at all, causing a poor exploration of the solution space.

Some methods are:

- **Single-point Crossover**

  In this method, a *crossover point* in the parents is randomly chosen. The bits to the right of the point are swapped to generate the offspring. This is directly related with human chromosomes crossover.

  Figure 2.7: Single-point Crossover example.

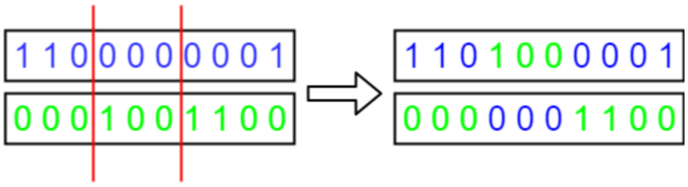
  

  Source: Elaborated by the Author.

- **k-point Crossover**

  The k-point method is an extension of the single-point one. Basically, k crossover points are randomly chosen, sorted and then the crossovers are performed. Thus, segments of the parents are swapped. k should be smaller than the size of the chromosomes.

  Figure 2.8: k-point Crossover example, with $k = 2$.

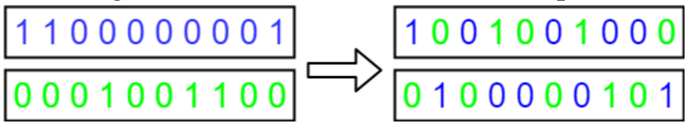
  

  Source: Elaborated by the Author.

- **Uniform Crossover**

  In this method, all offsprings' bits are randomly chosen from the parents'. An equal probability for each parent can be used, or other mixed formulas to reduce the randomness of this method. This should be used if there is no reason to inherit subsequent bits, *i.e.*, there is no logical relation between consecutive bits. This is the method used in this work.

  Figure 2.9: Uniform Crossover example.

  Source: Elaborated by the Author.

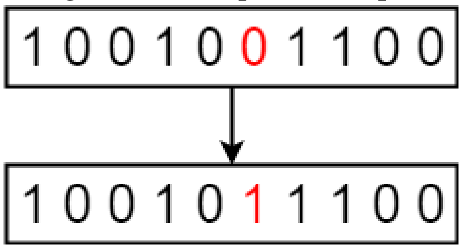
2.2.1.6 Mutation Methods

These methods introduce genetic diversity into the newly generated population. Mutation randomly changes one or more genes from the offsprings' chromosomes. It is equivalent to the genetic crossover that happens in biological sexual reproduction, which is proved as important for diversity in populations.

The chance of a bit being transformed is equal to the *mutation rate*. A number between 0 and 1 is chosen for each bit of each individual of the new population. If it is lower or equal to the mutation rate, the method is applied to the bit, otherwise the bit is left untouched. The mutation rate value should be really low, as a high value would turn the search into a primitive random search, which is not desired.

The method choice depends heavily on the application and the chromosome semantic. A common method is the **Flit Bit Mutation** for binary chromosomes, where the bit suffers a logical *NOT*, which is used in this work. For non-binary chromosomes, a common method is the **Gaussian Mutation**, where the bit has a new value randomly chosen considering a Gaussian distribution centered in the old value.

Figure 2.10: Flip Bit example.



Source: Elaborated by the Author.

2.2.1.7 Other Heuristics

GAs are highly customizable to the application it is being used to. Therefore, a wide range of different custom heuristics and methods can be applied in all steps of the algorithm.

One common example is **Elitism**, where a small subset of best individuals are carried from one generation to the next without any alteration. This guarantees the solution quality to not deteriorate and gives more chances to the other individuals to breed with the best one. The downside is that this can lead to local maxima, as this elite may represent this subset of solutions. It is used in this work, as further explained in the Proposal (Chapter 4).

Another common example is **Speciation**, where the crossover is changed to reduce crossover between two similar individuals, encouraging population diversity and avoiding local maxima by preventing early convergence. Nonetheless, this adds the need to define the individuals' similarity measure and demands more processing time.

2.2.1.8 Termination Criteria

There are many different criteria to define when to terminate the GA execution loop, and they can be combined. The most common are:

- A solution reaches a minimum fitness threshold (or the highest possible fitness);
- The highest fitness among individuals reached a long plateau, *i.e.*, is the same for n generations;
- A fixed number of generations was reached;
- Some kind of manual inspection of the individuals;
- Allocated limits are reached (space/time/computation cost).

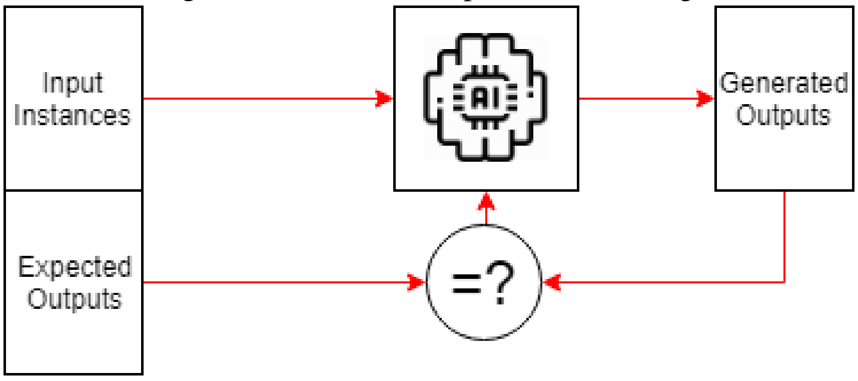
When choosing what combination of criteria to use and their values, the quality of the final answer should be considered: the less the algorithm runs, more areas of the search space will be left unexplored. Nonetheless, as any other AI method, the algorithm will converge and it doesn't make sense to search for solutions indefinitely. In this work, as it will be further explained in the Proposal (Chapter 4), we use a number of generations limit.

2.2.2 Supervised Learning

Supervised Learning is a type of Machine Learning problem where there is a dataset of observed **input instances**, *i.e.*, the **features**, which are labeled with the **expected outputs**, *i.e.*, the **classes** or **labels**, defined by an "oracle". For each input instance it receives, the model learns to generate the correct output.

There is a large number of algorithms designed to this kind of problems. Below, there are brief resumes of the ones used in this work. Their descriptions are based on knowledge acquired in AI classes during the course, which have Faceli *et al.* (2011) as main bibliography. They are all **classifiers**, which means that they predict a *label* for each input.

Figure 2.11: General Supervised Learning.



Source: Elaborated by the Author.

2.2.2.1 Gaussian Naïve Bayes

Naïve Bayes methods are a class of supervised learning algorithms that apply the Bayes Theorem:

$$P(A \mid B) = \frac{P(B|A)\,P(A)}{P(B)} \tag{2.2}$$

In the context of ML, A is the instance label y, and B is the feature vector $x_1, \ldots, x_n$, the formula can be rewritten as follows:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n|y)}{P(x_1, \ldots, x_n)} \tag{2.3}$$

The **Naïve** in the name means that these algorithms consider the *naive* assumption that all pairs of features are independent between themselves. This, with the math steps described in Pedregosa *et al.* (2018a), leads us to the final classification rule:

$$\begin{gathered} P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y) \\ \Downarrow \\ \hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i \mid y), \end{gathered} \tag{2.4}$$

In the *Gaussian* Naïve Bayes, we assume that the likelihood of the features is Gaussian, *i.e.*, if μ_y the mean of the values in the dataset associated with the class y, and σ_y is the variance of these same values, the following formula defines the likelihood of a feature x_i:

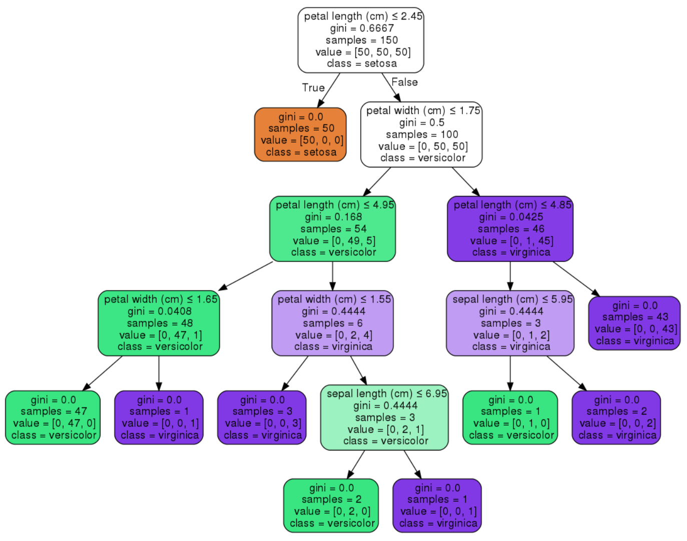
$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right) \tag{2.5}$$

2.2.2.2 Decision Tree

A Decision Tree is a model that defines how to classify the input given a set of simple if-then-else rules inferred from the data features. The decision tree building process is based on an entropy measurement process, where the impurity of the data is measured with the Gini index or the information gain measure and, if it passes a threshold, the data is separated in order to maximize the impurity reduction/information gain.

This is an example decision tree learned with this process to the Iris dataset:

Figure 2.12: Iris Dataset Decision Tree.



Source: (PEDREGOSA *et al.*, 2018b)

2.2.2.3 Random Forest

The Random Forest algorithm consists in learning a combination (ensemble) of **Decision Trees**, trained with some level of randomness in the dataset, giving different samples of the data for each Decision Tree. They are then combined with an average voting method. This method is specially strong against the Decision Trees tendency to overfit the data.

2.2.2.4 Quadratic Discriminant Analysis

The Quadratic Discriminant Analysis classifier tries to separate the instances of each class with a quadratic surface (the *quadratic decision boundary*), created with conditional densities of each class, using the Bayes' Rule, and the method assumes such densities are Gaussian. Unlike the Linear Discriminant Analysis method, not used in this work, this method doesn't assume that the covariance of the classes are identical.

2.2.2.5 Support Vector Machine

A Support Vector Machine creates a set of hyper-planes that divide the data by their classes, trying to achieve for each hyper-plane the highest distance to the nearest training data instances of any class. Thus, it creates a large functional margin, reducing the chance of error in the classifier when it is used with unknown data. In this work, the Radial Basis Function (RBF) kernel is used.

2.2.2.6 K-Nearest Neighbors

The K-Nearest Neighbors (KNN) Classifier is an algorithm that classifies a new instance based on the most common label of the K nearest already known instances. It *doesn't* build any type of model, using the training data itself to classify any new one.

The value of K is the key of this method. A low K means that only a few neighbors will be considered in the majority voting, making the algorithm more susceptible to noise, but with better classification boundaries. K should be an *odd* number, to avoid ties in the votings.

2.2.2.7 Logistic Regression

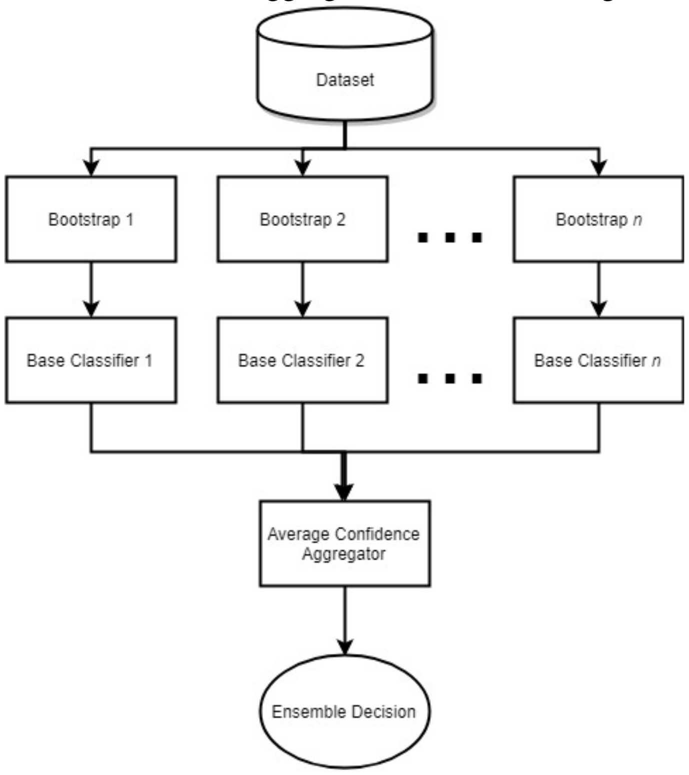
Despite its name, the Logistic Regression model tries to describe the probabilities of the possible outcomes with a logistic function. It can use a wide variety of algorithms to model the function, the one used in this work is a Coordinate Descendent algorithm.

2.2.3 Ensemble Learning

Ensemble Learning is the technique of combining classifiers to form a stronger one, *i.e.*, with a better predictive performance than any of the classifiers individually. The classifiers are trained separately, an their outputs are combined through some pre-defined strategy to generate the ensemble output. The two major types of ensembles are:

- **Homogeneous Ensemble Learning:** In this type of ensemble, all constituent classifiers have the same algorithm in their cores. The key here is how the information is distributed in the training phase between the different classifiers, so they learn different things and, together, they can choose the right answer in the testing phase, even though some of them fail due to the lack of complete information.
  The most common example of this type of Ensemble Learning is the **Random Forest** algorithm, that uses **Random Trees** as classifiers and combines their outputs with a *mode* operator (or an *average* operator for regression). Random Trees have high variance due to its randomness, and this is taken as advantage by the Random Forest, that train each tree with different partitions of the dataset. Also, as the classifiers doesn't have access to all information, it corrects the Random Trees tendency of overfitting.
  A common technique used here is **Bagging**. It is used to improve diversity in the ensemble by sampling the dataset for each basic classifier, doing a simple uniform sample with replacement, generating another one with the same size, with the intention of every classifier having a different view of the dataset. The voting is performed as a simple mode, or a confidence average of all basic algorithms to achieve better performances, to combine the different views of the data to avoid overfitting. This method is good with unstable high-variance algorithms, and is used in the Random Forest algorithm described above, but can degrade more stable ones, as it is depriving the algorithms of having access to all data, and they already have mechanisms to avoid overfitting.
- **Heterogeneous Ensemble Learning:** This type of ensemble is the opposite of the last one, as the constituent classifiers have different algorithms in their cores. Here, the most important thing is to explore the advantages and disadvantages of the different algorithms, combining them in a way that the strengths of some complement the weaknesses of the others, and vice versa. Concepts from homogeneous ensembles, such as dataset partitioning, can be used in the heterogeneous ensembles too, but that is not the main focus. This is the type used in our work.
  One example of this type of Ensemble Learning is the Voting Classifier, explained below.

Figure 2.13: Schematic of the Bagging method in an Homogeneous Ensemble.



Source: Elaborated by the Author.

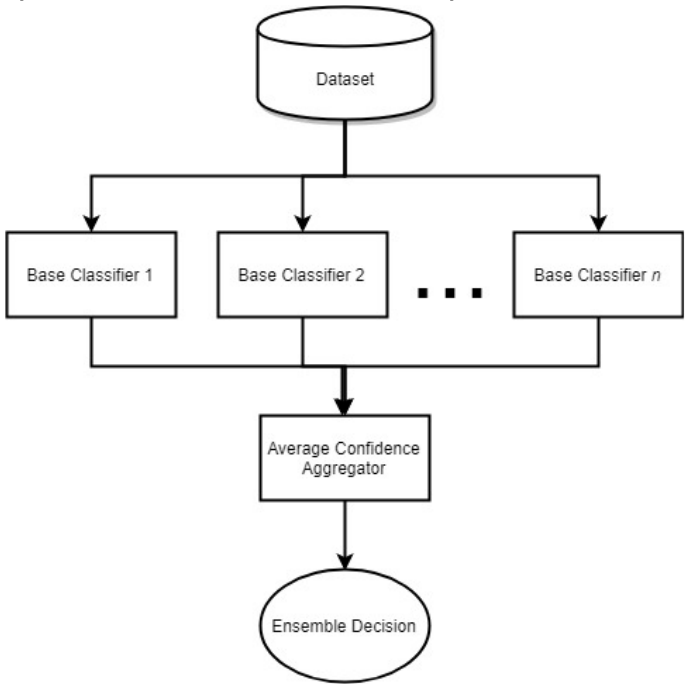
Ensemble Learning is a wide study area by itself, with a lot of variances. We will explain a small subset of possibilities that are relevant to this work.

2.2.3.1 Voting Classifier

The Voting Classifier is the most simple way to unite different classifiers. It is commonly used for heterogeneous groups of algorithms, as this doesn't include any type of dataset partitioning in the training phase, focusing on the differences between the constituent algorithms. In the most simple version of this method, all classifiers receive the full training dataset, training separately, and all their answers are taken in consideration in the voting without weighting, in a simple voting system, where the most common answer wins and is the output of the ensemble. A confidence average also can be used.

In this type of ensemble method, it is important to choose algorithms with complementary advantages and disadvantages, as the voting will consider all algorithms and therefore the downsides should be nullified. It is easy to customize due to its simplicity. For example, it is easy to add weights to the different classifiers in the voting, or to add some kind of dataset partitioning if wanted. The diversity is focused in the algorithms themselves, the method does nothing regarding it.

Figure 2.14: Schematic of the Voting Classifier method.



Source: Author

2.2.3.2 Diversity Measures

When building a ensemble, *i.e.*, a combination of classifiers, we rely on the voting between them to in fact classify the data. The classifiers themselves are weak, because of data sampling or their own weaknesses. In order to balance the weaknesses and strengths of the classifiers, so they are strong together, we need disagreement between them. Diversity between classifiers is recognized in the literature (CUNNINGHAM; CARNEY, 2000) as having an important role in the successful ensembles. Therefore, we should monitor how different are the algorithms classifying the instances and, if they are agreeing to much, our ensemble might not be performing as well as it could be.

Nonetheless, there are many ways to measure diversity, considering the number of classifiers that are being used. Kuncheva (2003) presents a collection of ten different measures. For ensembles with more than two classifiers, therefore *non-pairwise*, the most simple diversity measure is the **Entropy**, which is used in this work.

The Entropy comes from the Thermodynamics field (FRIGG; WERNDL, 2010), as many other ML concepts. It measures the randomness of a system, *i.e.*, how disordered it is. In Information Theory, as defined by Shannon (1948), the more two messages differ (*i.e.*, more entropy) the more information we gain.

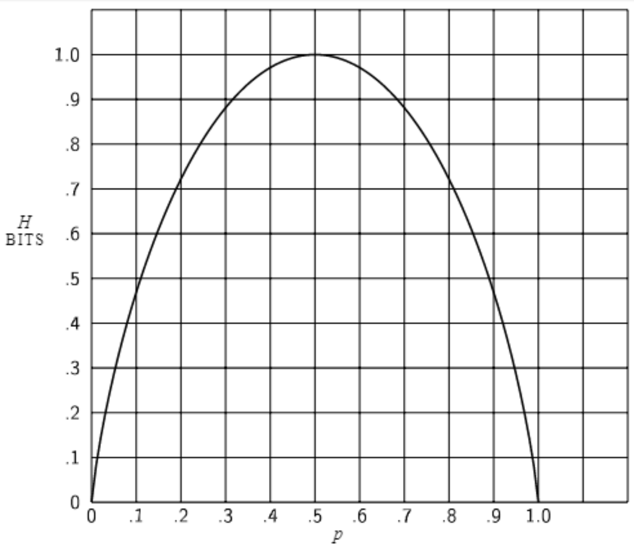
In ensembles, considering only two possible labels (0 and 1), an ensemble of L

classifiers has a maximum entropy if $\lfloor L/2 \rfloor$ classifiers agrees with one label and $\lfloor L - L/2 \rfloor$ with the other for a particular instance $z_j \in Z$. Denoting as $l(z_j)$ the number of classifiers from D that correctly classifies the instance z_j, a possible measure considering the Entropy concept is:

$$E = \frac{1}{N} \sum_{j=1}^{N} \frac{1}{(L - \lceil L/2 \rceil)} \min\{l(z_j), L - l(z_j)\} \tag{2.6}$$

The entropy value E varies between 0, that means complete agreement, and 1, that means maximum disagreement, *i.e.*, maximum diversity. The variation follows this parable, defined by Shannon:

Figure 2.15: Entropy H in the case of two possibilities with probabilities p and $(1 - p)$.



Source: (SHANNON, 1948).

2.2.4 Model Evaluation

In this subsection, we'll define the concept of **Cross Validation** and a set of metrics used in this work to measure the performance of the ensembles, therefore evaluating the models.

2.2.4.1 Cross Validation

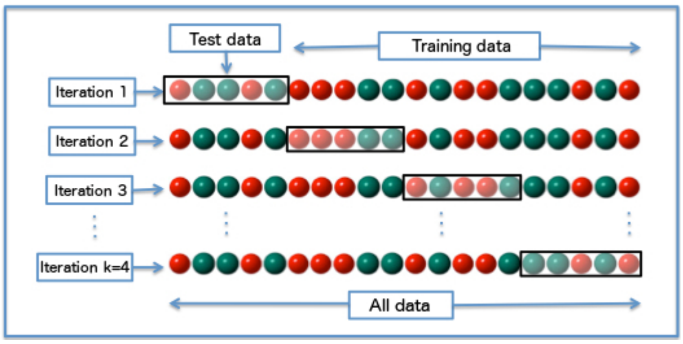
When training an ML model, we don't want it to memorize each instance of the dataset, and many algorithms can do it. When calculating performance measures in order to evaluate the generated models, if we use the same data we used to train the ML model, we'll be rewarding those which overfit the data.

As we want models with a good generalization level so it performs well when exposed to an independent dataset, this cannot happen. To consider it in the model evaluation, there is a class of methods called **Cross Validation**. They split the data in order to evaluate the model with data it hadn't access to when it was being trained. Of course, these methods always costs instances: we won't have all data to generate our AI model, which will make the learning process harder, especially if we don't have a large number of instances.

The most common is the **Holdout** method, where the dataset is divided into **train dataset** and **test dataset**. We use the former to train the AI, and the latter to test its performance. A common proportion is 80% of the data for training and 20% for testing, but this can be changed according to how large the original dataset is, considering that many algorithms does sampling by themselves.

Another method is the K**-fold Cross-Validation**, where we divide the data in K groups and train the model K times, with one of the groups not used in the training dataset in each training, being only used to test this iteration.

Figure 2.16: Diagram of a k-fold Cross Validation, with $k = 4$.



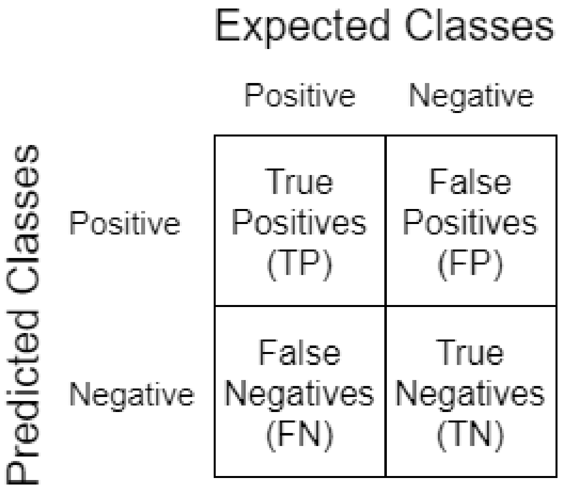
Source: (WIKIPEDIA, 2018).

The ***Stratified*** K***-fold Cross-Validation***, used in this work, selects the K groups preserving the percentage of samples for each class, so the original proportion of the classes is maintained. This is important for methods like Naïve Bayes, in which the *a priori* probabilities are a central part of the model design.

2.2.4.2 Confusion Matrix

The Confusion Matrix is a visualization of the ML model's predictions. The rows of the matrix are the predicted classes, and the columns are the expected classes, from the oracle. In each position of the matrix is the number of test instances classified according to the expected and the predicted class. The word *confusion* comes from the fact that, with this matricial visualization, it is easy to identify if the model is confusing the classes. We can also see if the model is classifying all instances to only one class, which is a common symptom of unbalanced datasets.

Figure 2.17: 2x2 Confusion Matrix



Source: Elaborated by the Author.

We can extract some performance measures from the Confusion Matrix. Below, are described the ones used in this work.

2.2.4.3 Accuracy

The most standard way to measure a model performance is by counting how many test instances were correctly classified in the entire test dataset, *i.e.*, the **Accuracy**. This measure is the most intuitive, but it rewards one-class classification. In terms of the Confusion Matrix, the formula to calculate the accuracy is the following:

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \tag{2.7}$$

2.2.4.4 F1 Score

The **F1 Score**, or **F-Measure**, is the harmonic mean of the **Precision** and the **Sensitivity** (or **Recall**) performance measures. They are defined as follows:

$$Precision = \frac{TP}{TP+FP} \tag{2.8}$$

$$Recall = \frac{TP}{TP+FN} \tag{2.9}$$

$$F1Score = \frac{Precision \times Recall}{Precision+Recall} = \frac{2TP}{2TP+FP+FN} \tag{2.10}$$

It is important to notice from the F1 Score formula that it doesn't consider the True Negatives (TN). It also rewards one-class classification.

2.2.4.5 Matthews Correlation Coefficient

The Matthews Correlation Coefficient, or **MCC**, is a measure first mentioned in Matthews (1975) that has the purpose of not rewarding models that classifies all instances as the majority class due to unbalancing. Unlike the other measures, it returns a value between -1 and 1, where -1 means complete failure, 0 means a performance equivalent to a random classifier (considering the proportion of the classes), and 1 the perfect classifier. It is the described by the following formula:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{2.11}$$

If any of the four sums equals zero, we define the denominator as 1 and, therefore, $MCC = 0$. It is claimed in the literature (CHICCO, 2017) as being the most informative measure for binary classifiers, specially in the case of unbalanced datasets.

2.2.4.6 Area Under Curve

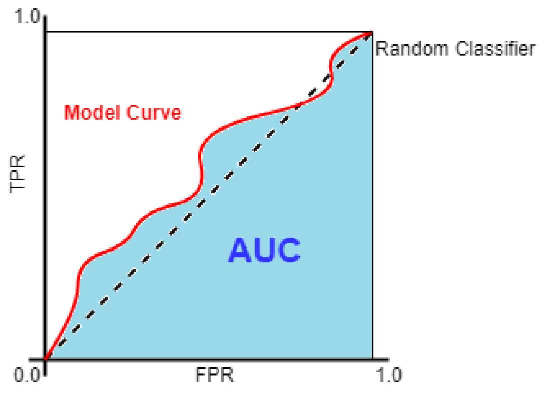
The **Area Under Curve**, or **AUC**, is a performance measure that uses the area under the **Receiver Operating Characteristic Curve**, or **ROC Curve**. The ROC Curve is the plot of the Recall, or True Positive Rate (TPR) against the False Positive Rate (FPR), that is equal to $1 - Specificity$, at various thresholds.

$$TPR = Recall = \frac{TP}{TP+FN} \tag{2.12}$$

$$FPR = \frac{FP}{FP+TN} = 1 - Specificity \tag{2.13}$$

The higher the curve compared to the diagonal (that represents a random classifier), the better. The AUC value varies between 0 and 1, 0 being the complete failure,

Figure 2.18: Area Under ROC Curve for a Model. A real model curve won't be curved, it will look like a ladder.



Source: Elaborated by the Author.

0.5 being the random classifier and 1 the perfect classifier. It is, assuming that positive instances are ranked higher than negative instances, "equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance." (FAWCETT, 2006, p. 868).

2.2.5 Unbalanced Datasets

A common problem in AI in general is class unbalancing in the data. It is natural that in the nature data isn't uniformly distributed, and datasets generated from real data will mirror this mismatch: a class may have more instances than others, because it occurs more in the observed ambient from which the dataset was generated. This can also happen due to the difficulty or ease in classifying instances from a given class, causing an unbalance.

The problem is: ML algorithms tend to classify all instances to the majority class if the unbalance is high. It is easier to do so and get, for example, 65% accuracy, then to try to learn something and lose accuracy in the process before achieving a higher one. Some algorithms tend to be lazy and conservative about changes, especially those which doesn't reinforce curiosity.

Regarding the dataset, there are two common methods to eliminate the unbalance for those algorithms which we can't simply generate a new balanced dataset from the source ambient: **downsampling** and **oversampling**.

- **Downsampling:** means sampling the majority class instances *without reposition* by the number of instances of the minority class. For example, if the majority class has 100 instances and the minority has 20, 20 instances from the majority class will be chosen and we'll have 40 instances in the final dataset, with 20 instances from each class.

The obvious problem with this method is that we're discarding data that should be useful for the learning process. Nonetheless, we are doing so to give all classes the same importance.

- **Oversampling:** does the opposite of downsampling. We sample the majority class *with reposition* until we have the same number of instances as the majority class. For example, if the majority class has 100 instances and the minority has 20, 100 instances from the minority class will be chosen with reposition (meaning we'll have repeated instances) and we'll have 200 instances in the final dataset, with 100 instances from each class.

  The problem here is that we're repeating some instances, what may introduce a false bias in the AI algorithm. Nevertheless, we are not discarding any data.

3 RELATED WORK

The scientific community is actively researching computational approaches to better understand miRNA mechanisms and how it participates in the metabolism and genetic diseases. Among these, ML algorithms have been incresingly used and are important to push the knowledge further based on manually curated datasets, providing useful information from which we can extract valuable insights and hypothesis regarding miRNA's action. Despite the significant advances, there is still a long way to go, with room for a lot of improvement in the prediction accuracy.

A common problem faced in the literature is the unbalance between the positive and the negative classes. Databases with experimentally validated miRNA-target interactions provide much more function pairs than non functional. This is partially due to the fact that the provided data is curated from the related literature and it is usually harder to publish negative results; therefore, these pairs are commonly discarded. In real life, indeed, non-target pairs are majority, leading to a big problem to ML model training.

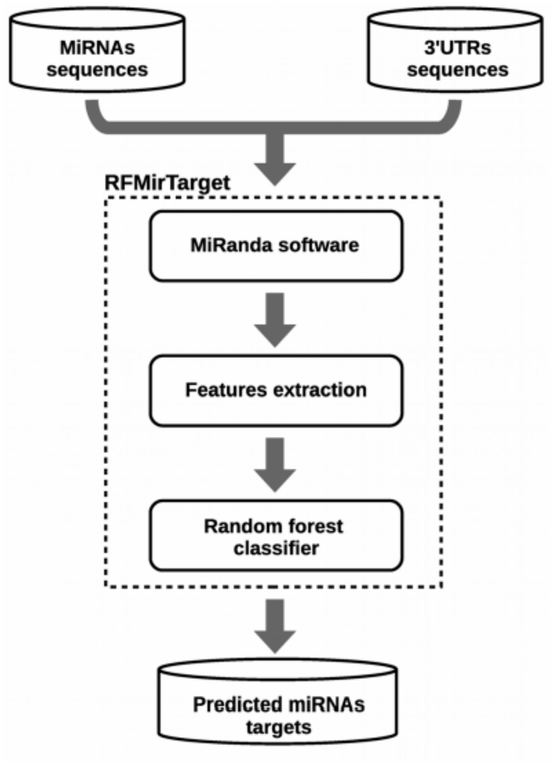
Ensemble learning variations are common among ML-based solutions for the miRNA-target prediction task due to their ability to tackle challenges such as small and unbalanced datasets, as well as harder classifications tasks. The idea of putting classifiers together to achieve better performances is explored by some articles.

3.1 Homogeneous Ensemble

Homogeneous ensemble has been explored in the work by Mendoza *et al.* (2013) and Yan *et al.* (2007). In the first publication, a Random Forest classifier was used, as their random feature selection and random instances sample could be used in order to circumvent the challenges of the miRNA target prediction task. A framework called RFMirTarget (Figure 3.1) was created, including the miRanda software for feature extraction from the miRNA and mRNA sequences, then using the Random Forest as ML model to generate the predictions, followed by a features importance analysis. The framework achieved an AUC score of 0.96, surpassing other approaches by the time of their publication.

Yan *et al.* (2007) also use an ensemble in their proposed algorithm. They generated a set of features with data from TarBase (VLACHOS *et al.*, 2015) and trained the ensemble with AdaBoost sampling and ten Support Vector Machine classifiers with a layer of feature selection to trim redundant and irrelevant generated features, using only

Figure 3.1: Mendoza *et al.* (2013) proposed framework, RFMirTarget.



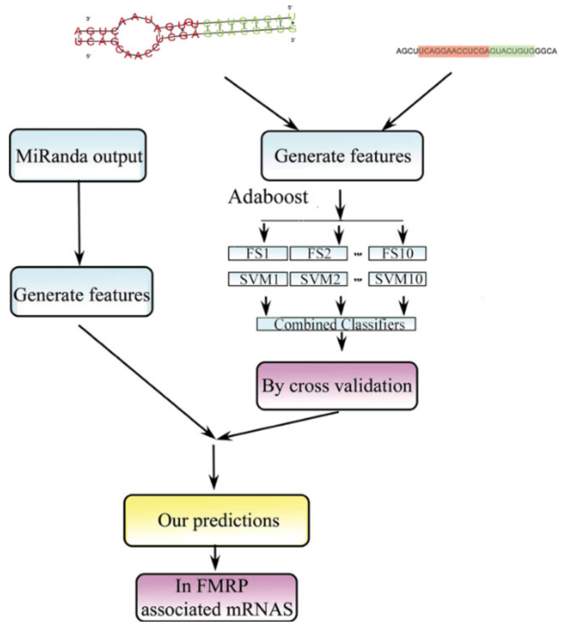
Source: (MENDOZA *et al.*, 2013).

the eight most relevant. The generated ensemble is then used with miRanda to create the predictions. The workflow is shown in Figure 3.2. According to the authors, the ensemble approach was used to avoid the unbalanced classes problem, without generating random instances, and achieved an accuracy of 82.95%.

3.2 Heterogeneous Ensemble

Yu *et al.* (2014) states that different prediction tools often present divergent sets of targets, conflicting in their predicted outputs. Therefore, if we combine those tools together in ensembles, their divergences converge into better results (Figure 3.3). They also state that "curating more instances of negative samples is also critical to further enhance the performance of the proposed approach" (YU *et al.*, 2014, p. 228), as "experimentalists are typically uninterested in collecting invalid miRNA–mRNA pairs and discard them" (YU *et al.*, 2014, p. 224), which causes the unbalancing in the datasets, creating a challenge to ML model training. Their proposed method consists in getting the classifications of six heterogeneous existing miRNA target prediction tools (TargetScan, miRanda, DIANA-microT, PITA, miRTarget2, and PicTar), processing the outcomes to generate

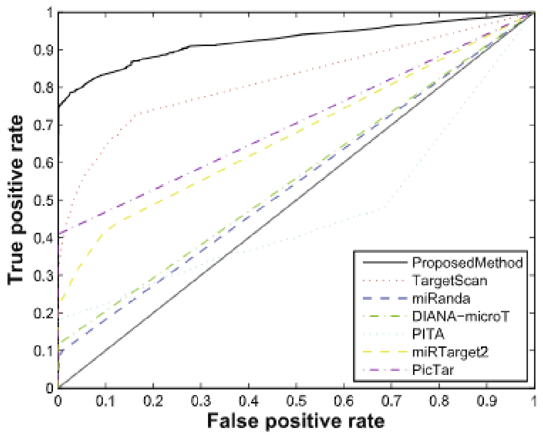
Figure 3.2: Yan *et al.* (2007) proposed workflow.



Source: (YAN *et al.*, 2007).

positive and negative training samples to a binary classifier. It's important to mention that they statistically generate negative samples to achieve a balanced dataset.

Figure 3.3: Comparison of Yu *et al.* (2014) proposed ensemble method against the tools used in the ensemble. It outperforms them by 52.5% in terms of AUC Score.



Source: (YU *et al.*, 2014).

Le *et al.* (2015) also demonstrate that an ensemble of different methods performs better than the methods alone in miRNA target prediction using expression data. In their publication, they combine several methods used for this task, *e.g.* Pearson's Correlation Coefficient (PCC) and Z-score (PRILL *et al.*, 2010), using a Borda count election (MARBACH *et al.*, 2012) to make the predictions. As illustrated in Figure 3.4, the combination of three methods (Pearson, IDA and Lasso) achieved the best result.

Figure 3.4: Le *et al.* (2015) ranking of different ensemble compositions against the individual methods.

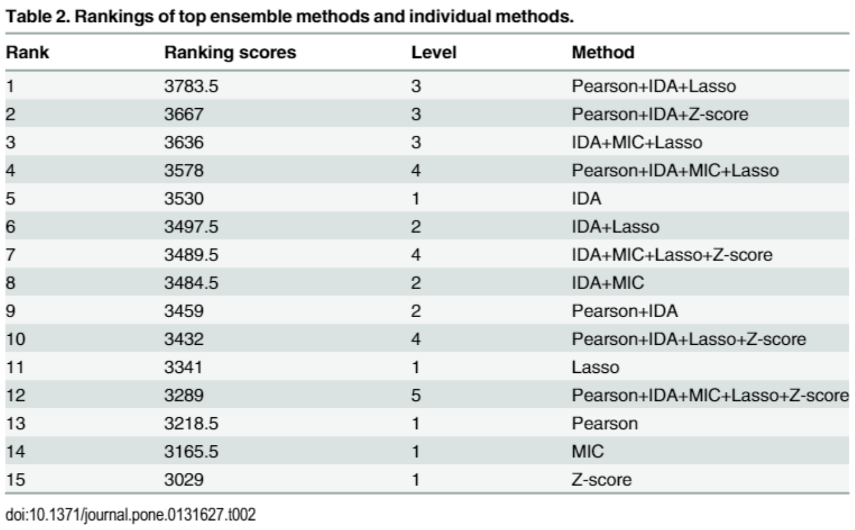


Table 2. Rankings of top ensemble methods and individual methods.

Rank	Ranking scores	Level	Method
1	3783.5	3	Pearson+IDA+Lasso
2	3667	3	Pearson+IDA+Z-score
3	3636	3	IDA+MIC+Lasso
4	3578	4	Pearson+IDA+MIC+Lasso
5	3530	1	IDA
6	3497.5	2	IDA+Lasso
7	3489.5	4	IDA+MIC+Lasso+Z-score
8	3484.5	2	IDA+MIC
9	3459	2	Pearson+IDA
10	3432	4	Pearson+IDA+Lasso+Z-score
11	3341	1	Lasso
12	3289	5	Pearson+IDA+MIC+Lasso+Z-score
13	3218.5	1	Pearson
14	3165.5	1	MIC
15	3029	1	Z-score

doi:10.1371/journal.pone.0131627.t002

Source: (LE *et al.*, 2015).

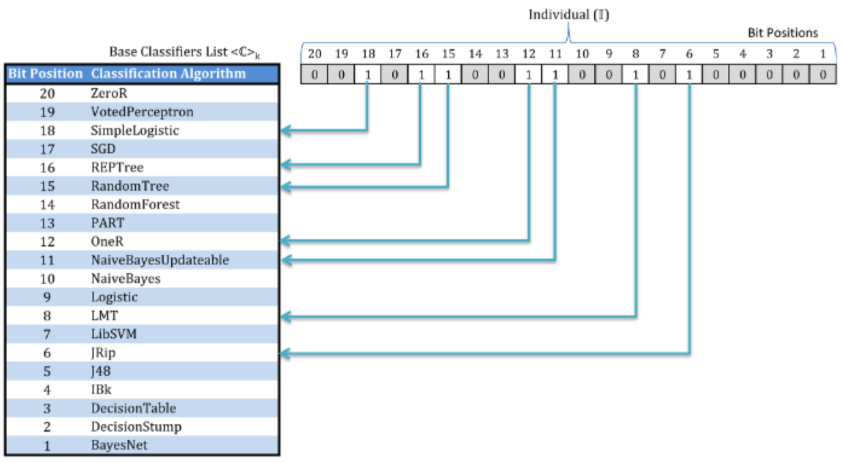
3.3 Genetic Algorithm with Heterogeneous Ensemble

The idea of searching for the best heterogeneous ensemble using a GA in the biology area has been previously explored by some publications. Haque *et al.* (2016) apply this concept in their GA-EoC (Genetic Algorithm - Ensemble of Classifiers) algorithm, as "the performance of an ensemble is dependent on the choice of constituent base classifiers" (HAQUE *et al.*, 2016, p. 1), using the GA to search for the best combination of classifiers would avoid an exhaustive search, which would have an exponential complexity.

GA-EoC uses in each ensemble a maximum of 20 base classifiers combined with a simple majority voting approach, with a random sub-sampling being used to balance the classes. An average of scores from 10-fold cross validation to evaluate the ensemble in terms of Matthews Correlation Coefficient (MCC) was used as fitness value in the GA. The Genetic Algorithm searches through the different combinations using Tournament Selection with 10 random individuals, Uniform Crossover with 60% crossover rate, and Flip-Bit Mutation with 0.01 mutation rate for each bit, plus one individual passed directly to the next population with Elitism. The population size was 100 and the representation of the ensemble in the genotype uses 0 or 1 for each algorithm: 0 meaning that the algorithm isn't used and 1 that it is used (Figure 3.5).

The approach was tested against seven datasets, from which three are from the UCI-ML repository, including the Wisconsin Breast Cancer (WBC) dataset, and one related with the Alzheimer Disease. Figure 3.6 shows that the GA-EoC algorithm had a superior performance against the individual algorithms, and Figure 3.7 shows that it had superior performance against other ensemble methods that include all individual algorithms.

Figure 3.5: Haque *et al.* (2016) genotype ensemble representation example.



Source: (HAQUE *et al.*, 2016).

Figure 3.6: Haque *et al.* (2016)'s GA-EoC comparison against the individual algorithms.

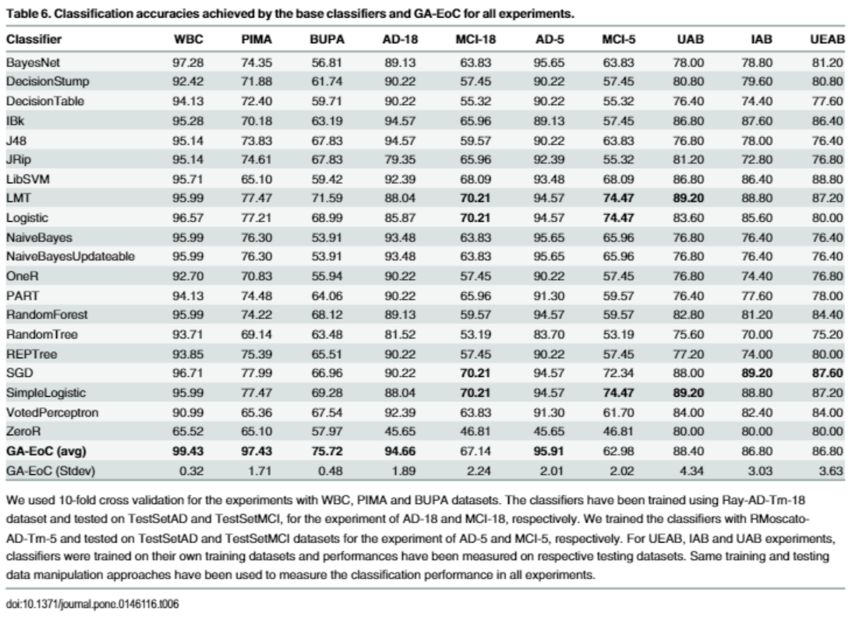


Table 6. Classification accuracies achieved by the base classifiers and GA-EoC for all experiments.

Classifier	WBC	PIMA	BUPA	AD-18	MCI-18	AD-5	MCI-5	UAB	IAB	UEAB
BayesNet	97.28	74.35	56.81	89.13	63.83	95.65	63.83	78.00	78.80	81.20
DecisionStump	92.42	71.88	61.74	90.22	57.45	90.22	57.45	80.80	79.60	80.80
DecisionTable	94.13	72.40	59.71	90.22	55.32	90.22	55.32	76.40	74.40	77.60
IBk	95.28	70.18	63.19	94.57	65.96	89.13	57.45	86.80	87.60	86.40
J48	95.14	73.83	67.83	94.57	59.57	90.22	63.83	76.80	78.00	76.40
JRip	95.14	74.61	67.83	79.35	65.96	92.39	55.32	81.20	72.80	76.80
LibSVM	95.71	65.10	59.42	92.39	68.09	93.48	68.09	86.80	86.40	88.80
LMT	95.99	77.47	71.59	88.04	**70.21**	94.57	**74.47**	**89.20**	88.80	87.20
Logistic	96.57	77.21	68.99	85.87	**70.21**	94.57	**74.47**	83.60	85.60	80.00
NaiveBayes	95.99	76.30	53.91	93.48	63.83	95.65	65.96	76.80	76.40	76.40
NaiveBayesUpdateable	95.99	76.30	53.91	93.48	63.83	95.65	65.96	76.80	76.40	76.40
OneR	92.70	70.83	55.94	90.22	57.45	90.22	57.45	76.80	74.40	76.80
PART	94.13	74.48	64.06	90.22	65.96	91.30	59.57	76.40	77.60	78.00
RandomForest	95.99	74.22	68.12	89.13	59.57	94.57	59.57	82.80	81.20	84.40
RandomTree	93.71	69.14	63.48	81.52	53.19	83.70	53.19	75.60	70.00	75.20
REPTree	93.85	75.39	65.51	90.22	57.45	90.22	57.45	77.20	74.00	80.00
SGD	96.71	77.99	66.96	90.22	**70.21**	94.57	72.34	88.00	**89.20**	**87.60**
SimpleLogistic	95.99	77.47	69.28	88.04	**70.21**	94.57	**74.47**	**89.20**	88.80	87.20
VotedPerceptron	90.99	65.36	67.54	92.39	63.83	91.30	61.70	84.00	82.40	84.00
ZeroR	65.52	65.10	57.97	45.65	46.81	45.65	46.81	80.00	80.00	80.00
GA-EoC (avg)	**99.43**	**97.43**	**75.72**	**94.66**	67.14	**95.91**	62.98	88.40	86.80	86.80
GA-EoC (Stdev)	0.32	1.71	0.48	1.89	2.24	2.01	2.02	4.34	3.03	3.63

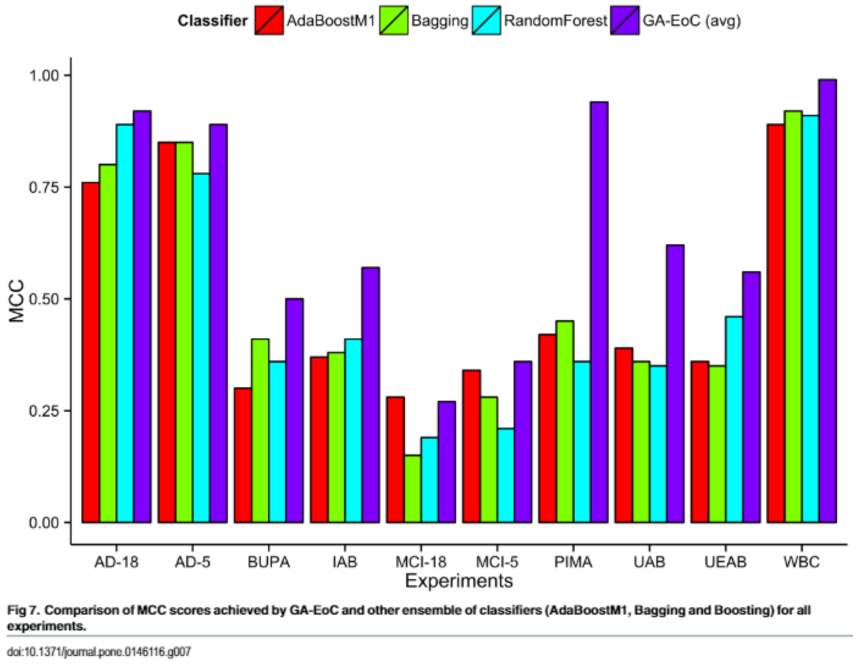
We used 10-fold cross validation for the experiments with WBC, PIMA and BUPA datasets. The classifiers have been trained using Ray-AD-Tm-18 dataset and tested on TestSetAD and TestSetMCI, for the experiment of AD-18 and MCI-18, respectively. We trained the classifiers with RMoscato-AD-Tm-5 and tested on TestSetAD and TestSetMCI datasets for the experiment of AD-5 and MCI-5, respectively. For UEAB, IAB and UAB experiments, classifiers were trained on their own training datasets and performances have been measured on respective testing datasets. Same training and testing data manipulation approaches have been used to measure the classification performance in all experiments.

doi:10.1371/journal.pone.0146116.t006

Source: (HAQUE *et al.*, 2016).

In the miRNA target prediction problem, Mousavi, Eftekhari and Haghighi (2015) used, in their proposed EP-RTF algorithm, a GA to perform an ensemble pruning to choose the classifiers to be trained using the Rotation Forest algorithm, being its parameter K also optimized by the GA (named M in the article, being the number of subsets of features in the Rotation Forest algorithm). The classifiers' predictions are combined with

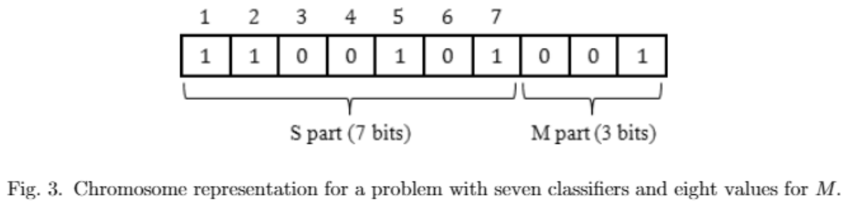
Figure 3.7: Haque *et al.* (2016)'s GA-EoC comparison against other ensemble methods.



Source: (HAQUE *et al.*, 2016).

weighted majority voting. The individual representation can be seen in Figure 3.8.

Figure 3.8: Mousavi, Eftekhari and Haghighi (2015) genotype ensemble representation example. Each chromosome encodes a subset of classifiers and one value of M.


1 2 3 4 5 6 7
1 1 0 0 1 0 1 0 0 1
S part (7 bits)
M part (3 bits)
Fig. 3. Chromosome representation for a problem with seven classifiers and eight values for M.

Source: (MOUSAVI; EFTEKHARI; HAGHIGHI, 2015).

The Rotation Forest (RODRÍGUEZ; KUNCHEVA; ALONSO, 2006) is a state-of-art ensemble method that focuses in feature extraction to encourage accuracy and diversity in the ensemble. The algorithm divides the features randomly in K subsets (K is a parameter, mentioned above) and applies Principal Component Analysis on each of them to maintain the main components. After this, K axis rotations are performed to generate the training dataset to the basic classifier. This method is good when losing features is not a problem, and the parameter K needs to be manually defined. As this is a newer and more complex method than the others, it is harder to customize it to specific necessities.

A list of 13 basic classifiers (including Naïve Bayes, Feed-Forward Back Propagation Neural Network, Support Vector Machine and K-Nearest Neighbors) were used in the GA proposed by Mousavi, Eftekhari and Haghighi (2015), and according to their observations, "it is clear that structurally different base classifiers (*i.e.* heterogeneous ensemble) are employed for creating diversity in the ensemble" (MOUSAVI; EFTEKHARI; HAGHIGHI, 2015, p. 10). Figure 3.9 shows that the EP-RTF algorithm had a superior performance against the individual algorithms, and Figure 3.10 shows that it had superior performance against other ensemble methods that include all individual algorithms (*i.e.*, without optimization of ensembles composition).

Figure 3.9: Mousavi, Eftekhari and Haghighi (2015)'s EP-RTF comparison against the individual algorithms.

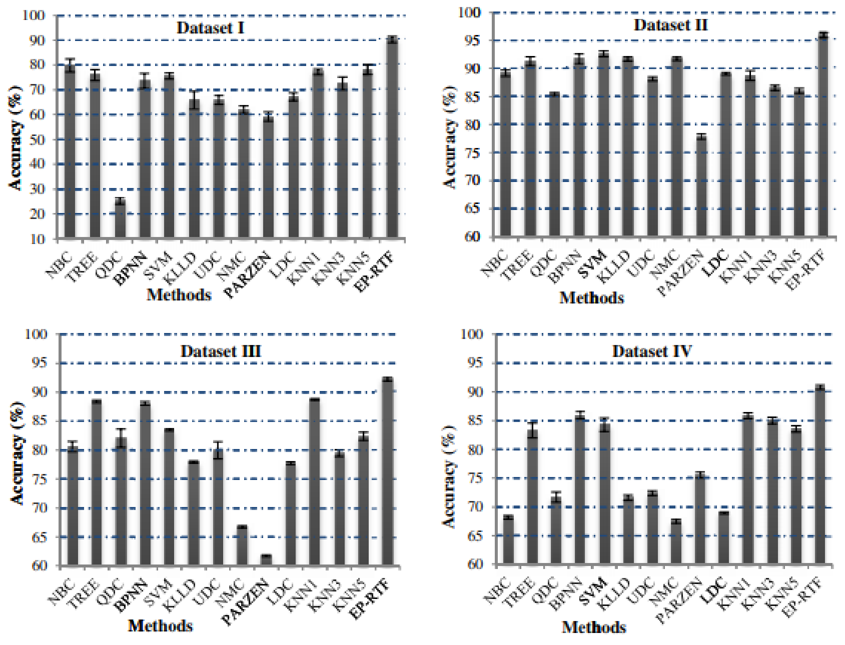


Fig. 5. Comparison of average accuracy with the corresponding standard deviation over the four human miRNAs target prediction datasets of the 14 methods.

Source: (MOUSAVI; EFTEKHARI; HAGHIGHI, 2015).

Figure 3.10: Mousavi, Eftekhari and Haghighi (2015)'s EP-RTF comparison against other ensemble methods.

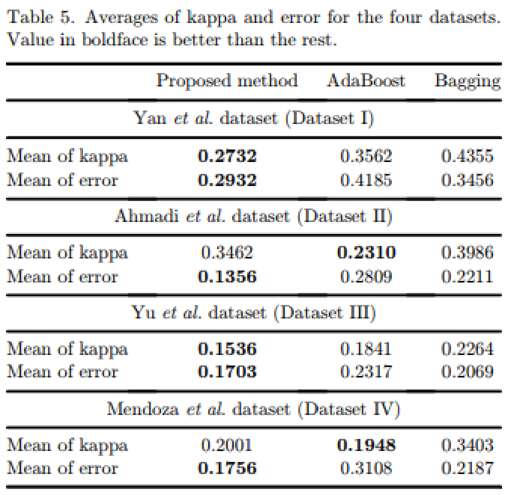


Table 5. Averages of kappa and error for the four datasets. Value in boldface is better than the rest.

	Proposed method	AdaBoost	Bagging
Yan *et al.* dataset (Dataset I)			
Mean of kappa	**0.2732**	0.3562	0.4355
Mean of error	**0.2932**	0.4185	0.3456
Ahmadi *et al.* dataset (Dataset II)			
Mean of kappa	0.3462	**0.2310**	0.3986
Mean of error	**0.1356**	0.2809	0.2211
Yu *et al.* dataset (Dataset III)			
Mean of kappa	**0.1536**	0.1841	0.2264
Mean of error	**0.1703**	0.2317	0.2069
Mendoza *et al.* dataset (Dataset IV)			
Mean of kappa	0.2001	**0.1948**	0.3403
Mean of error	**0.1756**	0.3108	0.2187

Source: (MOUSAVI; EFTEKHARI; HAGHIGHI, 2015).

4 PROPOSED SOLUTION

The purpose of this work is to evaluate the influence of using explicit measures of diversity in the fitness value along with performance metrics in the overall quality of the ensembles. In the next sections, each component of the adopted methodology will be explained in detail.

4.1 Data Sources

4.1.1 miRNA-Targets Dataset

To create the dataset to be used by the ensembles, a large data source is needed with positive and negative examples of miRNA-mRNA pairs, ideally with similar quantities for both classes. Sadly, as already stated in the Related Work section (Section 3), there isn't a very large dataset. They are only limited to thousands of examples for each class, where in reality there are more than millions of possible combinations. Furthermore, the datasets are largely unbalanced, having much more positive examples than negative ones. Considering this, a dataset that minimizes these problems shall be chosen.

The two most used miRNA-targets datasets are miRTarBase (CHOU *et al.*, 2018) and DIANA-TarBase (VLACHOS *et al.*, 2015; KARAGKOUNI *et al.*, 2018). These databases use different methods to curate the literature, but are both composed by experimentally validated miRNA-mRNA pairs for a variety of species, including humans. Both are works in progress, and released new versions this year. Nonetheless, previous stable versions were considered for this work (*i.e.*, miRTarBase v6.1 and DIANA-TarBase v7.0).

As presented in Table 4.1, DIANA-TarBase v7.0 has a larger total number of samples, and considerably less unbalancing between the classes, still being high nonetheless. Therefore, this dataset was chosen for the development of the current work.

Table 4.1: Positive and negative examples in miRTarBase v6.1 and DIANA-TarBase v7.0, the latter being used in the current work.

	Positive/Functional	Negative/Non-Functional	Total
miRTarBase v6.1	6958 *(96.1%)*	283 *(3.9%)*	7241
DIANA-TarBase v7.0	**5619** ***(74.3%)***	**1944** ***(25.7%)***	**7563**

Source: Elaborated by the Author.

4.1.2 miRNA and mRNA Sequences

DIANA-TarBase provides the list of miRNAs and their respective targets represented by their official identifiers. However, to extract features for the miRNA target prediction, nucleotide sequences are needed. Therefore, miRNA and mRNA (*i.e.*, target gene) sequences must be collected from other sources.

For miRNA sequences, were gathered from miRBase (GRIFFITHS-JONES, 2006; KOZOMARA; GRIFFITHS-JONES, 2014) a database with more than 30,000 miRNA sequences from more than 200 species. It was also used in Mendoza *et al.* (2013) and other related works.

For mRNA sequences, we have chosen the BioMart portal Smedley *et al.* (2015), a unified interface to access to more than 800 different databases. It is important to mention that in this source, the same mRNA can have different versions (for example, the *POU2F2* mRNA), which means different sequences. This is due to bilogical aspects related to the DNA transcription and transcript post-processing, whose details are out of the scope of this work. We have considered all the different transcripts related to a given gene.

4.2 Dataset Generation

First, we combined the different raw sequences retrieved for DIANA-TarBase entries to generate miRNA-target positive and negative examples for further processing. A simple script was developed using R to perform this task, generating a total of 23,019 positive examples and 7,855 negative examples. These negative examples comprise 25.44% of non-functional samples collected from DIANA TarBase. The growth in the examples happened due to the multiple versions of the sequences in the BioMart mRNA sequences dataset. We removed examples that appeared in both positive and negative classes, to avoid conflicts.

Next, we used the 2010 version of the miRanda software (ENRIGHT *et al.*, 2003) to perform sequence alighment between miRNA and target gene. The default configurations were used when running the software. In this phase, a lot of the extra examples from the previous phase were removed, as they were invalid. Specifically, 72.46% of the miRNA-mRNA combinations (71.72% positive and 74.67% negative) were lost, including, 57.22% of the miRNAs represented in the original dataset (56.01% positive and 60.76% negative). Thus, from this phase we have obtained 7,100 positive and 2,131 negative dataset examples (*i.e.*, 23.08% examples are in the negative class), being 2,441 different miRNAs in the positive examples and 746 different miRNA in the negative ones (*i.e.*, 23.08% of the miRNAs are in the negative class), having 2.85 valid miRNA-gene examples per miRNA in the positive class, and 2.91 in the negative class.

With these examples in hands, we extracted the features of each combination with a Perl script to parse the output from miRanda software. The features set used in this work was previously proposed by Mendoza *et al.* (2013) (Table 4.2). Below, there is a table resuming the numbers detailed above (Table 4.3).

Table 4.2: Features used in this work, based on Mendoza *et al.* (2013)'s features for RFMirTarget.

#	**Feature Name**	#	**Feature Name**
1	Alignment Score (by miRanda)	18	Position 10
2	Alighment Length	19	Position 11
3	Minimum free energy of the alignment	20	Position 12
4	G:C's absolute frequency in the alignment	21	Position 13
5	A:U's absolute frequency in the alignment	22	Position 14
6	G:U's absolute frequency in the alignment	23	Position 15
7	Number of gaps in the alignment	24	Position 16
8	Number of mismatches in the alignment	25	Position 17
9	Position 1	26	Position 18
10	Position 2	27	Position 19
11	Position 3	28	Position 20
12	Position 4	29	Minimum free energy of the seed
13	Position 5	30	G:C's absolute frequency in the seed
14	Position 6	31	A:U's absolute frequency in the seed
15	Position 7	32	G:U's absolute frequency in the seed
16	Position 8	33	Number of gaps in the seed
17	Position 9	34	Number of mismatches in the seed

Source: Elaborated by the Author.

Table 4.3: Progression of positive and negative examples.

	miRTarBase	Combinations	**Valid Combinations**	Different miRNA
Positive/Functional	5,619 *(74.3%)*	23,019 *(74,6%)*	**7,100 *(76.9%)***	2441
Negative/Non-Functional	1,944 *(25.7%)*	7,855 *(25.4%)*	**2,131 *(23.1%)***	746
Total	7,563	30,874	**9,231**	-[1]

Source: Elaborated by the Author.

[1] - There is an intersection between miRNA in Positive and Negative classes.

4.3 Ensemble's Base Classifiers and Aggregation Function

Eleven different classifiers were chosen from the *scikit-learn* Python package (PEDREGOSA *et al.*, 2011) to compose the ensemble, whose calls are displayed in Table 4.4. The methods were chosen based on the works by Mousavi, Eftekhari and Haghighi (2015, p. 10) and Haque *et al.* (2016, p. 5), both containing high-variance classifiers.

Table 4.4: Classifiers used in the ensemble.

#	Classifier Name	Classifier *scikit-learn* Call
1	Gaussian Naïve Bayes	*GaussianNB()*
2	Decision Tree (Gini index, $max\_depth = 5$)	*DecisionTreeClassifier(max_depth=5, criterion='gini')*
3	Decision Tree (Entropy, $max\_depth = 5$)	*DecisionTreeClassifier(max_depth=5, criterion='entropy')*
4	Random Forest (Gini index, $max\_depth = 5$)	*RandomForestClassifier(max_depth=5, criterion='gini')*
5	Random Forest (Gini index, $max\_depth = 5$)	*RandomForestClassifier(max_depth=5, criterion='entropy')*
6	Quadratic Discriminant Analysis	*QuadraticDiscriminantAnalysis()*
7	Support Vector Machine	*SVC(kernel='rbf', probability=True)*
8	K-Nearest Neighbors ($K = 3$)	*KNeighborsClassifier(n_neighbors=3)*
9	K-Nearest Neighbors ($K = 5$)	*KNeighborsClassifier(n_neighbors=5)*
10	K-Nearest Neighbors ($K = 7$)	*KNeighborsClassifier(n_neighbors=7)*
11	Logistic Regression	*LogisticRegression()*

Source: Elaborated by the Author.

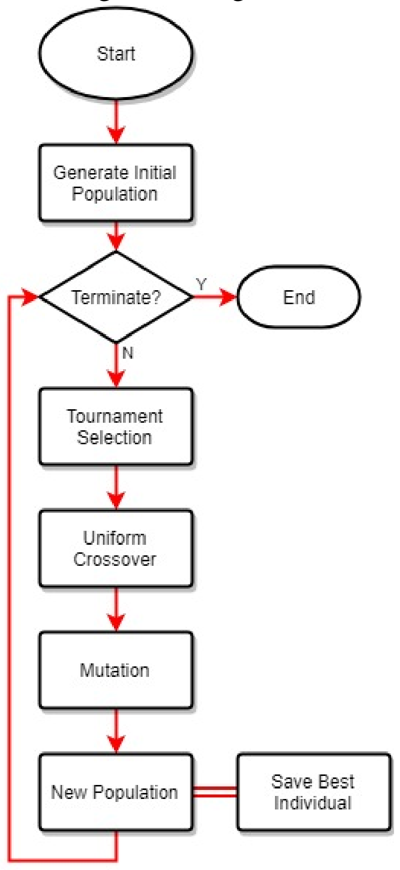
The methods were combined using a Voting Classifier (implemented by the author, described at Section 2.2.3.1), meaning that all classifiers were trained using the full training dataset, without any sampling technique. The outputs were generated with *predict_proba()*, which return probability estimates to the given inputs, and combined with a simple average. If the output is lower than 0.5, the instance is classified as Negative; otherwise, it is classified as Positive.

Discarded classifiers and other techniques (*e.g.*, downsampling), and why they weren't used in the final version of this work, are mentioned in the Discarded Variations section (Section 4.5).

4.4 Genetic Algorithm for Ensemble Optimization

The adopted Genetic Algorithm follows a basic high level execution, illustrated in Figure 4.1 below. Each phase details will be explained in the following subsections, such as hyperparameters configuration, also listed in Table 4.5, and algorithms used.

Figure 4.1: Genetic Algorithm high level execution pipeline.



Source: Elaborated by the Author.

Table 4.5: Adopted GA's hyperparameters configuration.

Hyperparameter Name	Hyperparameter Value
Population Size	55
Crossover Rate	60%
Mutation Rate	1%
Elitism Size	1
Tournament Size	Population Size$/10 = 5$
Generations Limit	10

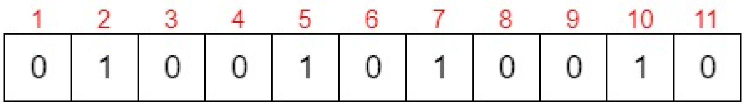
Source: Elaborated by the Author.

4.4.1 Chromosome - Ensemble Representation

Every individual in the GA represents a different combination of classifiers in the ensemble. The chromosome defines for each base classifier if it is used or not in the ensemble, with the number 1 meaning $true$ (*i.e.*, it is used), and the number 0 meaning $false$ (*i.e.*, it is not used). If any chromosome is fully composed of zeroes, which has a probability of only $1/2^{11}$ to happen in the initial generation, and even lower during the GA execution (as a ensemble with a low number of classifiers won't have good fitness), a random new chromosome is generated.

Considering the numeration in the first column of Table 4.4, Figure 4.2 shows a possible chromosome, *i.e.*, a classifiers combination in an ensemble, and Figure 4.3 illustrates the ensemble construction and evaluation process based on the chromosome/genotype.

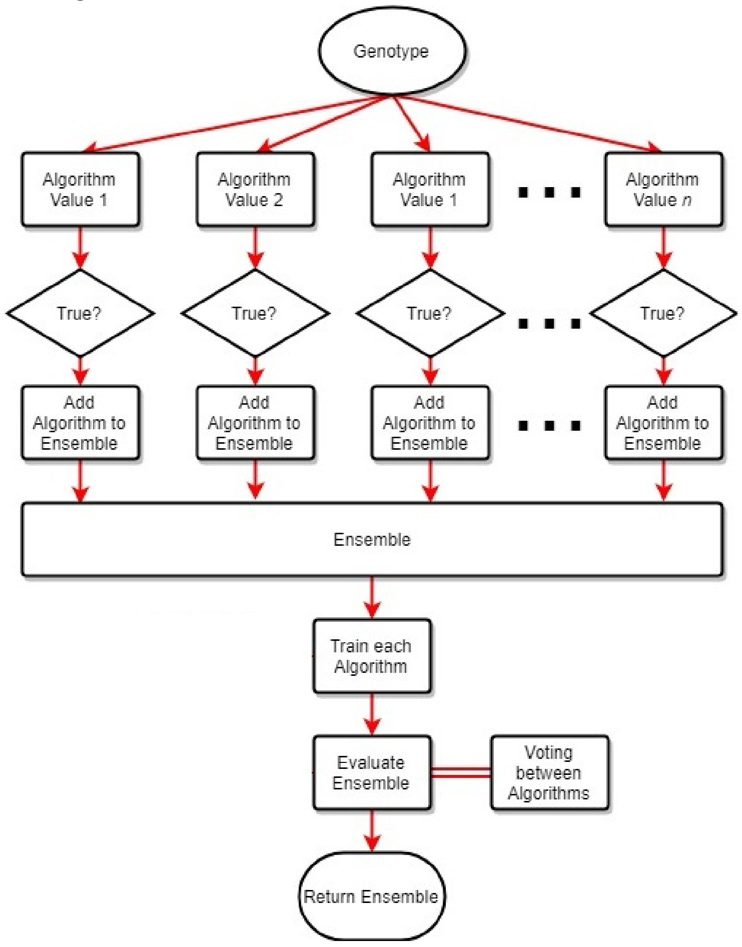
Figure 4.2: Example chromosome representing an ensemble that uses classifiers $\#2$, $\#5$, $\#7$, and $\#10$ from Table 4.4.



1	2	3	4	5	6	7	8	9	10	11
0	1	0	0	1	0	1	0	0	1	0

Source: Elaborated by the Author.

Figure 4.3: Ensemble construction and evaluation.



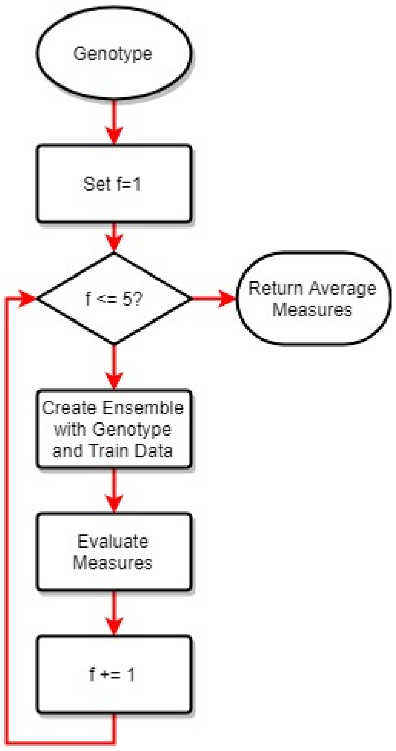
Source: Elaborated by the Author.

4.4.2 Fitness

Each individual is evaluated taking the average of a Stratified K-Fold Cross Validation (Section 2.2.4.1, Figure 4.4) of the ensemble, with $K = 5$, using one of the performance measures (Accuracy, Matthews Correlation Coefficient, F1 Score, Area Under the Curve) explained in Section 2.2.4. These measures are combined or not with the Entropy diversity measure explained in Section 2.2.3.2, with weights (β and $(1-\beta)$) depending on the experiment being executed, with the following formula:

$$fitness(x) = \beta \times Performance(x) + (1-\beta) \times Diversity(x) \tag{4.1}$$

Figure 4.4: Stratified K-Fold Cross Validation ensemble evaluation.



Source: Elaborated by the Author.

4.4.3 Population Size

The population size was chosen according to a formula used in Haque *et al.* (2016), described by Cox (2005), being k the size of the chromosome, and 2^k the number of possible ensemble combinations:

$$population\_size = min\left((5 \times k), \left(\frac{2^k}{2}\right)\right) \tag{4.2}$$

As we have $k = 11$, the result of the above formula is $population\_size = 55$.

4.4.4 Offspring Generation

The offspring, or new population, is generated from a combination of chromosomes from individuals in the previous generation (*i.e.*, parents). As explained in Section 2.2.1, this phase has three steps: Selection, Recombination (Crossover), and Mutation.

The Selection method used in this work is the **Tournament Selection**, explained in Section 2.2.1.4. The $size$ of the tournament, *i.e.*, the number of individuals randomly chosen for the selection, is $population\_size/10$, and $p = 1$ is used to enforce maximum selection pressure. This means that the best individual has a high chance of being chosen, and this probability decays for each subsequent individual in the ranking. This also causes the 10% worst individuals to be mathematically discarded because they won't be the top individual of the tournament they enter, as we use the *deterministic* variation of the method, forcing at least a 10% refresh of the population.

We also apply the **Elitism** strategy, explained in Section 2.2.1.7, with $n = 1$. The best individual is always added to the next population without any change.

The method used in this work for Recombination is the **Uniform Crossover**, explained in Section 2.2.1.5. We have chosen this method as there isn't any meaning in the order of the classifiers in the chromosome, and we want to maximize the combination of solutions that work. Given two individuals selected by Tournament Selection, they have a chance of 60% of being recombined via Uniform Crossover to generate their offspring. If they are not recombined, they move on to the mutation step as they are.

For mutation, we applied the **Flip-Bit Mutation** method, explained in Section 2.2.1.6, with a mutation chance of 1% per bit in the chromosome, leading to a $1 - 0.99^{11} = 10.47\%$ probability of a chromosome of size 11 to be mutated at least one time.

4.4.5 Implementation and Optimizations

The implementation was made in *Python 3.6*, using the packages *numpy* for float number operations, *pandas* for dataset management, *scipy* for scientific operations, and *sklearn* (or *scikit-learn*) for Machine Learning methods, *e.g.* the classifiers used in the ensemble.

There were two major practical optimizations in the implementation of this Genetic Algorithm. As we don't have a low chance of individuals to be maintained from a generation to another (*i.e.*, $40\% \times (100\% - 10.47\%) = 35,81\%$), and the ensemble evaluation process takes a considerable amount of time, a *cache* was implemented to prevent us of executing the same ensemble combination more than once.

The other optimization made consists of saving the full individual object, containing the trained ensemble inside of it, of the best individual of each generation, thinking of a possible future use of encountered ensembles. Saving is made using a Python library called *Pickle*, that serializes the object into a binary file that can be recovered again in a future moment using the same library.

The full implementation is provided at <https://github.com/skakim/miRNA_predict>.

4.5 Discarded Variations

During the implementation, we've had some discarded variations that are worth mentioning in this section. The reasons why each one of them were abandoned are described in the next subsections.

4.5.1 Rotation Forest and Bagging

A implementation of the Rotation Forest technique described in the related work by Mousavi, Eftekhari and Haghighi (2015) (Section 3.3) was the first ensemble approach we have tried, together with Bagging. Both methods' purpose is to present different data to the different classifiers that compose the ensemble. Rotation Forest does this by sampling the features, and Bagging by sampling the instances.

Both approaches were outperformed by the Voting Classifier. The hypothesis for the Bagging to be worst is that, as mentioned in the Biological Background (Section 2.1) and in the miRNA-targets Dataset (Section 4.1.1), the number of miRNA-mRNA pairs available in the dataset is incredibly low compared to the number of possible combinations. Therefore, any instances sample will deprive the classification of examples even more, degrading their effectiveness.

The hypothesis for the poor performance of the Random Forest algorithm is similar, and also has a basis in the Biological Background (Section 2.1): we don't know exactly which features are relevant to classify a miRNA-mRNA pair as functional. Feature sampling conducted by the algorithm may forbid the classifiers to discover it by themselves, not letting them have access to all the relevant information and therefore also degrading their performance.

4.5.2 Downsampling and Oversampling

Thinking about the unbalancing of the classes in the dataset described in the Dataset Generation (Section 4.2), the most common techniques to mitigate its effect is downsampling and oversampling, explained in Section 2.2.5. It is important to mention that, in this context, as we don't understand the mechanisms behind miRNA-mRNA interactions, we can't generate artificial instances to the minority class, therefore we are tied to the oversampling and downsampling techniques.

Similar to the ensemble methods described in the previous section, the techniques didn't perform well. The hypothesis for the poor performance of downsampling is the same as Bagging: sampling the already low number of instances will cause the classifiers to have even less information, degrading their capacity of learning.

On the other hand, our hypothesis for the weak performance of oversampling is based on the way the DIANA-TarBase database (and others) is generated. The examples are taken from published articles, which have a tendency to publish only positive examples, as noticed in Section 4.1.1. The few articles that publish negative examples are usually restricted to a few miRNAs or diseases of higher scientific interest, thus generating a sample that is not fully representative of the universe of non-functional miRNA-target genes. If we oversample this class, any existing bias will be replicated, causing even more problems to the classifiers, and a worse performance compared to not using any sampling technique.

With those results, we've decided to rely on the heterogeneity of the classifiers in the ensemble to create the variety of models and the desired diversity that is the core of the ensemble methods.

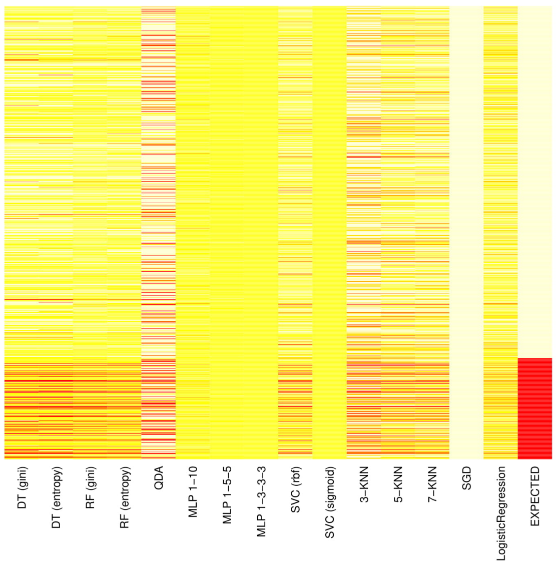
4.5.3 Discarded Classifiers

During the process of selecting the base classifiers, there were some methods that performed badly with any combination of parameters tried, and therefore were removed from the list. They are:

- Neural Networks, adopting the following topologies of hidden layers: 1-10, 1-5-5, 1-3-3-3, 10, 15 and 20, always with a maximum number of iterations equal to 50;
- Support Vector Machine with Sigmoid Kernel;
- Stochastic Gradient Descent.

The hypothesis for the bad performance is that these algorithms suffer too much with class unbalancing, achieving a poor generalization power. In Figure 4.5, generated during the evaluation of the methods to be used in the final version, it is clear that those algorithms weren't learning, since they were always classifying instances as belonging to one specific class or always generating the same output.

Figure 4.5: Heatmap of classifiers predicted probabilities for the positive class for instances in a test dataset. Probabilities closer to 0.0 are shown in red, whereas probabilities closer to 1.0 are represented in light yellow. The column "EXPECTED" provides the true label, in which red means negative examples and light yellow denotes positive examples.



Source: Elaborated by the Author.

5 EXPERIMENTAL RESULTS

The proposed solution was extensively tested with different combinations of performance measures and with different proportions of the diversity measure (entropy). We've also compared the performance of the ensemble optimized with a GA against the individual classifiers and the full ensemble (composed of a majority voting among all classifiers). The results are explained and analyzed in the following sections.

5.1 Computational Resources

Some dataset generation code and all the GA code was made with *Python 3.6.3*. The exact versions of the external Python libraries used are as follows:

- *numpy* - 1.13.3
- *pandas* - 0.20.3
- *pickleshare* - 0.7.4
- *scikit-learn* - 0.19.1

Another part of the dataset generation code was made with *R 3.5.0* and *Perl 5.26.1*, and was executed in an *Ubuntu 18.04.1* Virtual Machine.

The GA execution was made in a *Windows 10* computer with an *Intel Core i5-7400 @ 3.00GHz* CPU and 8GB RAM. A GA execution uses, on average, 300KB of RAM, only one CPU core, as the implemented Voting Classifier doesn't use parallelism, and finishes the processing of the 10th generation in approximately 45 minutes. The majority of this time is used in the first GA generations.

5.2 Experiment Methodology

For experimentation, we've used the parameters explained in the Proposed Solution (Chapter 4), summarized in Table 4.5.

Although we collect all performance measures for each ensemble, in the following sections we compare the different combinations using the *AUC* measure, given its sensitivity to overfitting, easy interpretation, and variability observed in the tests. The complete table of performance measures is available in Appendix A.

As explained in the Proposal (Subsection 4.4.2), the measures of an ensemble are obtained through an average of the 5-Fold Stratified Cross-Validation. As the whole process involves a lot of random events, to evaluate the performance-diversity proportions in the GA's fitness, each combination was executed 10 times to extract the following results. The fitness' function combinations are:

- Pure Accuracy
- 75% Accuracy + 25% Diversity
- 50% Accuracy + 50% Diversity
- 25% Accuracy + 75% Diversity
- Pure AUC Score
- 75% AUC Score + 25% Diversity
- 50% AUC Score + 50% Diversity
- 25% AUC Score + 75% Diversity
- Pure F1 Measure
- 75% F1 Measure + 25% Diversity
- 50% F1 Measure + 50% Diversity
- 25% F1 Measure + 75% Diversity
- Pure MCC
- 75% MCC + 25% Diversity
- 50% MCC + 50% Diversity
- 25% MCC + 75% Diversity

5.3 Genetic Algorithm Learning Curves

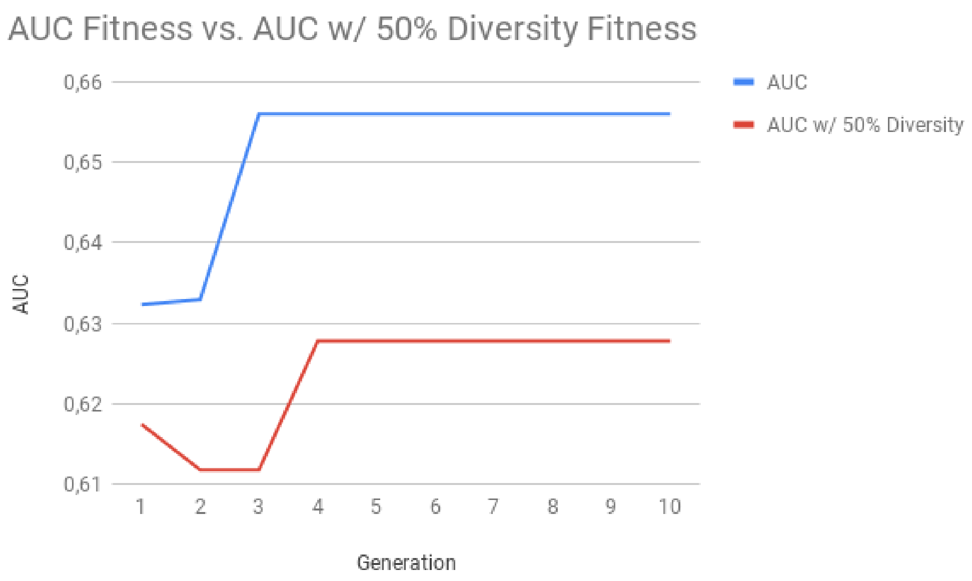
A common visualization to evaluate the learning process of an AI model is the learning curve. It shows, for each learning step, what is the achieved performance. Good learning curves normally have logarithmic forms, *i.e.*, the model learns fast in the beginning and progresses slowly towards the optimal solution until it stabilizes.

Figure 5.1 presents the learning curve for a particular execution of the proposed method with two different combinations of performance and diversity in the fitness function: pure AUC *versus* AUC combined with Diversity with similar weight (50% each).

It is possible to notice that the fitness function based on the combination of AUC and Diversity leads to a slight drop in the performance in the second generation, as it isn't using the pure performance measure as fitness function, but it recovers in the fourth generation and stabilizes.

All combinations stabilize in the first generations, with rare exceptions. This may be caused by a variety of factors: the amount of selection pressure caused by the Tournament Selection, Elitism and, most importantly, the unbalancing of the dataset. There is a limit where the classifiers stop trying to learn how to properly classify the minority class, leading to an early stabilization as the ceiling is reached. What the GA is finding here is the best way to combine different classifiers in order to have a better performance

Figure 5.1: Learning curves for fitness curves using pure AUC and AUC combined with 50% of diversity. Performed is compared by means of AUC Score.



Source: Elaborated by the Author.

by averaging the outputs of the saturated classifiers, *i.e.*, learning how they complement each other better.

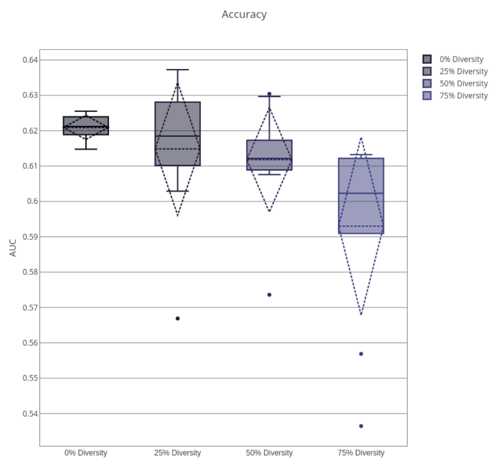
5.4 Comparison between Different Diversity Proportions

The comparison of different proportions of performance and diversity measures is visualized as boxplots in Figure 5.2. As mentioned before, each boxplot is the result of ten executions of the GA learning pipeline. The dotted diamonds overlapping the boxplots show the standard deviation, whereas its center line represents the average value. The different GA's fitness functions are compared in terms of AUC Score. Other performance comparisons lead to similar conclusions. We've chosen this one as it has produced a good visualization. To read the other measured performances, please refer to Appendix A.

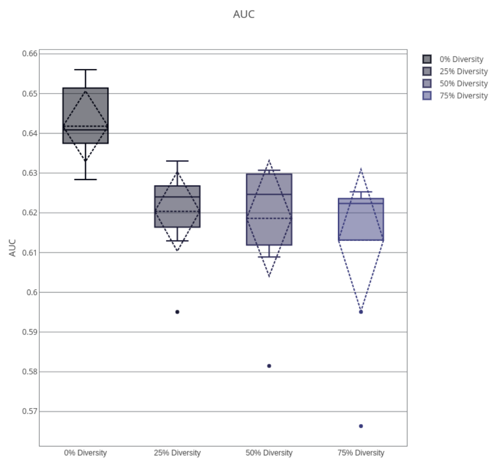
It is noticeable, given the visualization, that the more we increase the proportion of diversity in the fitness, the higher is the standard deviation observed and the more outliers we have. Nonetheless, it is also possible to perceive that a dose of diversity can also produce good outliers, *i.e.*, better solutions than a pure performance fitness.

Considering the challenge presented by the unbalanced dataset and the produced plots, we believe that the balancing difficulty can be overshadowing the benefits of using diversity in the fitness, as it saturates rapidly. With a larger and balanced dataset, diversity

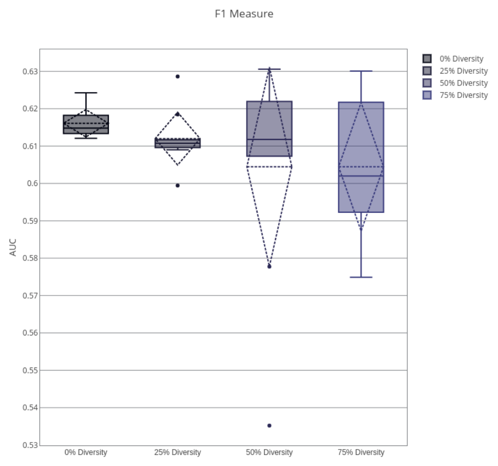
Figure 5.2: Boxplots comparing the performance in terms of AUC Score (Y axis) for every combination of performance measure with different proportions of the diversity measure (X axis) in the GA's fitness functions specified in the plots' titles. Results are extracted from 10 repetitions of the proposed solution.



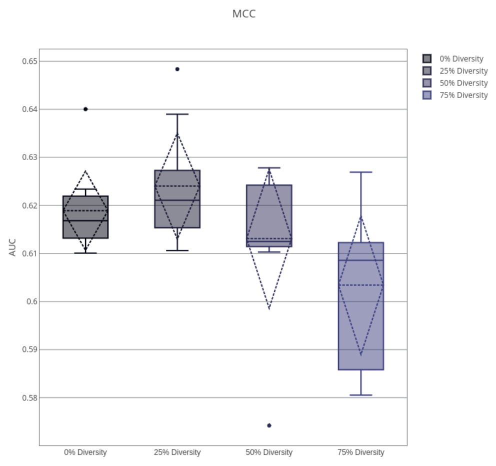
(a) Accuracy



(b) AUC



(c) F1 Measure



(d) MCC

Source: Elaborated by the Author.

may positively affect the performance of the Genetic Algorithm, but this can't be concluded from the tests executed in this work. Additionally, we can't conclude that diversity measure isn't useful in the miRNA-target prediction problem, as in some scenarios a GA fitness function explicitly optimizing diversity achieves better solutions.

5.5 Comparison against Individual Classifiers and Full Ensemble

In this section, we provide experimental results that compare the proposed GA approach against the individual classifiers used in the ensembles, and the ensemble using all possible classifiers. Table 5.1 shows the average AUC scores for each approach, being the 5-Fold Stratified Cross-Validation average for individual classifiers and the full ensemble, and the average of 10 executions for the GA executions.

Table 5.1: Average AUC Score for each approach.

Approach	AUC Score
Gaussian Naïve Bayes	0.5943 ± 0.0106
Decision Tree (Gini index, $max\_depth = 5$)	0.6182 ± 0.0111
Decision Tree (Entropy, $max\_depth = 5$)	**0.6220 ± 0.0075**
Random Forest (Gini index, $max\_depth = 5$)	0.6110 ± 0.0096
Random Forest (Entropy, $max\_depth = 5$)	0.6108 ± 0.0082
Quadratic Discriminant Analysis	0.5868 ± 0.0382
Support Vector Machine	0.6054 ± 0.0135
K-Nearest Neighbors ($K = 3$)	0.6084 ± 0.0195
K-Nearest Neighbors ($K = 5$)	0.6058 ± 0.0198
K-Nearest Neighbors ($K = 7$)	0.5961 ± 0.0165
Logistic Regression	0.5365 ± 0.0062
Full Ensemble	0.6107 ± 0.0110
Genetic Algorithm (AUC w/o Diversity)	**0.6418 ± 0.0094**
Genetic Algorithm (AUC w/ 25% Diversity)	0.6204 ± 0.0105
Genetic Algorithm (AUC w/ 50% Diversity)	0.6186 ± 0.0153
Genetic Algorithm (AUC w/ 75% Diversity)	0.6131 ± 0.0188

Source: Elaborated by the Author.

The Genetic Algorithm outvalues the individual classifiers and the full ensemble. When using combinations of diversity in the GA's fitness function, the average AUC Scores are lower than the one from the best individual algorithm (Decision Tree with Entropy), but it is important to notice that the standard deviation is larger. Combinations of just a few classifiers tend to have better fitness values than those with a lot of classifiers, and this tendency can be justified by this table: the full ensemble is one of the worst approaches.

It is interesting to mention that, even though some algorithms are perceptibly worse in terms of AUC Score performance, they are eventually chosen by the Genetic Algorithm to compose a solution, as the most important in a heterogeneous ensemble is to have complementarity between its base classifiers.

6 CONCLUSION

From the results shown in the previous chapter, we can conclude that using an Genetic Algorithm to build an heterogeneous ensemble for the miRNA-target prediction problem produces better ensembles than simply using all possible classifiers, or using them alone. This is true even when comparing against the Random Forest technique, that is a homogeneous ensemble technique.

Even though the implementation was a success, we can’t conclude from the results obtained if adding a diversity measure to the GA fitness function increases its performance of not. The class unbalancing in the dataset, along with the relatively small sample size, presented a big challenge to the proposed technique, and may have overshadowed the benefits of the diversity measure. The higher variance when using the entropy in the GA fitness function leaded to some ensembles that are better than those generated with a pure performance fitness, but also generated worse ones.

Further work on this topic is needed to have more concrete conclusions. A larger and more balanced miRNA-mRNA dataset is strongly required for a new execution of the proposed solution. Other possible line of work, while we don’t have the desired dataset, would be to explore newer state-of-art multi-objective GA approaches, such as NSGA-II (DEB *et al.*, 2000), to automatically balance the performance with the diversity, by considering the Pareto-optimal front, in order to find better solutions that achieve both objectives. Summing up, further study is needed, extending our experimental approach with a larger and more balanced dataset to be able to effectively evaluate the proposed method.

REFERENCES

MENDOZA, M. R. *et al.* RFMirTarget: Predicting Human MicroRNA Target Genes with a Random Forest Classifier. **PLoS ONE**, v. 8, n. 7, 2013. ISSN 19326203.

YAN, X. *et al.* Improving the prediction of human microRNA target genes by using ensemble algorithm. **FEBS Letters**, 2007. ISSN 00145793.

YU, S. *et al.* Ensemble learning can significantly improve human microRNA target prediction. **Methods**, 2014. ISSN 10959130.

LE, T. D. *et al.* Ensemble methods for miRNA target prediction from expression data. **PLoS ONE**, 2015. ISSN 19326203.

HAQUE, M. N. *et al.* Heterogeneous ensemble combination search using genetic algorithm for class imbalanced data classification. **PLoS ONE**, 2016. ISSN 19326203.

MOUSAVI, R.; EFTEKHARI, M.; HAGHIGHI, M. G. A new approach to human MicroRNA target prediction using ensemble pruning and rotation forest. **Journal of bioinformatics and computational biology**, v. 13, n. 6, p. 1550017, 2015. ISSN 1757-6334. Available at: <http://www.ncbi.nlm.nih.gov/pubmed/26017463>.

FILIPOWICZ, W.; BHATTACHARYYA, S. N.; SONENBERG, N. **Mechanisms of post-transcriptional regulation by microRNAs: Are the answers in sight?** 2008.

YUE, D.; LIU, H.; HUANG, Y. Survey of Computational Algorithms for MicroRNA Target Prediction. **Current Genomics**, 2009. ISSN 13892029.

BARTEL, D. P. **MicroRNAs: Genomics, Biogenesis, Mechanism, and Function**. 2004.

HE, L.; HANNON, G. J. **MicroRNAs: Small RNAs with a big role in gene regulation**. 2004.

YANAIHARA, N. *et al.* Unique microRNA molecular profiles in lung cancer diagnosis and prognosis. **Cancer Cell**, 2006. ISSN 15356108.

KLUIVER, J. *et al.* BIC and miR-155 are highly expressed in Hodgkin, primary mediastinal and diffuse large B cell lymphomas. **The Journal of Pathology**, 2005. ISSN 00223417.

ZAHA, A.; FERREIRA, H. B.; PASSAGLIA, L. M. P. **Biologia Molecular Básica**. 5. ed. Porto Alegre: ArtMed, 2014. 416 p.

CRICK, F. Central dogma of molecular biology. **Nature**, 1970. ISSN 00280836.

CLANCY, S.; BROWN, W. Translation: DNA to mRNA to Protein. **Nature Education**, 2008. Available at: <https://www.nature.com/scitable/topicpage/translation-dna-to-mrna-to-protein-393>.

LU, M. *et al.* An analysis of human microRNA and disease associations. **PLoS ONE**, 2008. ISSN 19326203.

STURM, M. *et al.* TargetSpy: A supervised machine learning approach for microRNA target prediction. **BMC Bioinformatics**, 2010. ISSN 14712105.

VLACHOS, I. S. *et al.* DIANA-TarBase v7.0: Indexing more than half a million experimentally supported miRNA:mRNA interactions. **Nucleic Acids Research**, v. 43, n. D1, p. D153–D159, 2015. ISSN 13624962.

DARWIN, C. **On the Origin of Species**. London: John Murray, 1859.

TURING, A. M. Computing machinery and intelligence. n. 236, 1950. Available at: <http://mind.oxfordjournals.org/>.

FRASER, A. S. Simulation of Genetic Systems by Automatic Digital Computers. **Australian Journal of Biological Sciences**, 1957.

HORNBY, G. *et al.* Automated Antenna Design with Evolutionary Algorithms. In: **Space 2006**. [S.l.: s.n.], 2006. ISBN 978-1-62410-049-9. ISSN 1063-6560.

FACELI, K. *et al.* **Inteligência artificial: uma abordagem de aprendizado de máquina**. 1. ed. [S.l.: s.n.], 2011. 394 p.

PEDREGOSA, F. *et al.* **Naive Bayes**. 2018. Accessed in 14/10/2018. Available at: <http://scikit-learn.org/stable/modules/naive_bayes.html#naive-bayes>.

PEDREGOSA, F. *et al.* **Decision Trees**. 2018. Accessed in 14/10/2018. Available at: <http://scikit-learn.org/stable/modules/tree.html>.

CUNNINGHAM, P.; CARNEY, J. **Diversity versus Quality in Classification Ensembles based on Feature Selection**. [S.l.], 2000.

KUNCHEVA, L. I. **Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy**. [S.l.], 2003. v. 51, 181–207 p.

FRIGG, R.; WERNDL, C. **Entropy-A Guide for the Perplexed**. [S.l.], 2010.

SHANNON, C. E. **A Mathematical Theory of Communication**. [S.l.], 1948. v. 27, 623–656 p.

WIKIPEDIA. **Cross-validation (statistics)**. 2018. Accessed in 15/10/2018. Available at: <https://en.wikipedia.org/wiki/Cross-validation_(statistics)>.

MATTHEWS, B. W. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. **BBA - Protein Structure**, p. 442–451, 1975.

CHICCO, D. **Ten quick tips for machine learning in computational biology**. 2017. 1–17 p.

FAWCETT, T. An introduction to ROC analysis. **Pattern Recognition Letters**, p. 861–874, 2006.

PRILL, R. J. *et al.* Towards a rigorous assessment of systems biology models: The DREAM3 challenges. **PLoS ONE**, 2010. ISSN 19326203.

MARBACH, D. *et al.* Wisdom of crowds for robust gene network inference. **Nature Methods**, 2012. ISSN 15487091.

RODRÍGUEZ, J. J.; KUNCHEVA, L. I.; ALONSO, C. J. Rotation forest: A New classifier ensemble method. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 28, n. 10, p. 1619–1630, 2006. ISSN 01628828.

CHOU, C. H. *et al.* MiRTarBase update 2018: A resource for experimentally validated microRNA-target interactions. **Nucleic Acids Research**, Oxford University Press, v. 46, n. D1, p. D296–D302, 2018. ISSN 13624962.

KARAGKOUNI, D. *et al.* DIANA-TarBase v8: A decade-long collection of experimentally supported miRNA-gene interactions. **Nucleic Acids Research**, Oxford University Press, v. 46, n. D1, p. D239–D245, 2018. ISSN 13624962.

GRIFFITHS-JONES, S. miRBase: microRNA sequences, targets and gene nomenclature. **Nucleic Acids Research**, 2006. ISSN 0305-1048.

KOZOMARA, A.; GRIFFITHS-JONES, S. MiRBase: Annotating high confidence microRNAs using deep sequencing data. **Nucleic Acids Research**, 2014. ISSN 03051048.

SMEDLEY, D. *et al.* The BioMart community portal: An innovative alternative to large, centralized data repositories. **Nucleic Acids Research**, 2015. ISSN 13624962.

ENRIGHT, A. J. *et al.* **MicroRNA targets in Drosophila**. [S.l.], 2003. v. 5, n. 1, 1 p. Available at: <http://genomebiology.com/2003/5/1/R1>.

PEDREGOSA, F. *et al.* **Scikit-learn: Machine Learning in Python**. [S.l.], 2011. v. 12, 2825–2830 p. Available at: <http://scikit-learn.sourceforge.net.>

COX, E. **Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration**. 1. ed. San Francisco, CA: Elsevier/Morgan Kaufmann, 2005. 540 p. ISBN 9780121942755.

DEB, K. *et al.* A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-Objective Optimization. **Proceedings of the Parallel Problem Solving from Nature VI Conference, Paris, France**, p. 849–858, 2000.

APPENDIX A — COMPLETE TABLES WITH ALL EXPERIMENTAL EXECUTIONS OF THE PROPOSED SOLUTION

AUC	1	2	3	4	5	6	7	8	9	10	Avg	StdDev	Min	Max
Accuracy	0,8081	0,8011	0,8040	0,8178	0,8196	0,7993	0,8160	0,8030	0,8024	0,8154	0,8087	0,0077	0,7993	0,8196
AUC	0,6395	0,6422	0,6375	0,6439	0,6284	0,6514	0,6286	0,6560	0,6519	0,6386	0,6418	0,0094	0,6284	0,6560
F1	0,8840	0,8785	0,8807	0,8908	0,8934	0,8757	0,8909	0,8778	0,8786	0,8886	0,8839	0,0065	0,8757	0,8934
MCC	0,3690	0,3552	0,3637	0,3956	0,3883	0,3683	0,3745	0,3870	0,3691	0,3800	0,3751	0,0125	0,3552	0,3956
AUC + Diversity (25%)	1	2	3	4	5	6	7	8	9	10	Avg	StdDev	Min	Max
Accuracy	0,8034	0,7420	0,8209	0,8201	0,8229	0,8249	0,8203	0,7595	0,8200	0,8168	0,8051	0,0295	0,7420	0,8249
AUC	0,6330	0,6164	0,6198	0,6247	0,6267	0,6130	0,6270	0,5951	0,6242	0,6238	0,6204	0,0105	0,5951	0,6330
F1	0,8813	0,8352	0,8951	0,8941	0,8959	0,8983	0,8941	0,8522	0,8940	0,8918	0,8832	0,0217	0,8352	0,8983
MCC	0,3481	0,2393	0,3916	0,3923	0,4011	0,4139	0,3903	0,2217	0,3909	0,3782	0,3567	0,0688	0,2217	0,4139
AUC + Diversity (50%)	1	2	3	4	5	6	7	8	9	10	Avg	StdDev	Min	Max
Accuracy	0,8013	0,7835	0,7910	0,6820	0,8227	0,8209	0,7741	0,8202	0,7336	0,8168	0,7846	0,0455	0,6820	0,8227
AUC	0,6282	0,6215	0,6119	0,6089	0,6307	0,6299	0,6160	0,6278	0,5815	0,6297	0,6186	0,0153	0,5815	0,6307
F1	0,8801	0,8678	0,8741	0,7798	0,8955	0,8943	0,8611	0,8939	0,8318	0,8914	0,8670	0,0365	0,7798	0,8955
MCC	0,3407	0,2918	0,3001	0,2047	0,4015	0,3939	0,2710	0,3901	0,1774	0,3782	0,3149	0,0801	0,1774	0,4015
AUC + Diversity (75%)	1	2	3	4	5	6	7	8	9	10	Avg	StdDev	Min	Max
Accuracy	0,8186	0,6681	0,8219	0,7774	0,7410	0,8221	0,7735	0,8217	0,8198	0,8201	0,7884	0,0509	0,6681	0,8221
AUC	0,6246	0,5663	0,6236	0,6147	0,5951	0,6253	0,6132	0,6235	0,6228	0,6218	0,6131	0,0188	0,5663	0,6253
F1	0,8930	0,7433	0,8954	0,8638	0,8371	0,8955	0,8610	0,8984	0,8940	0,8944	0,8676	0,0483	0,7433	0,8984
MCC	0,3920	0,1522	0,4001	0,2730	0,2024	0,3971	0,2658	0,3921	0,3975	0,3888	0,3261	0,0944	0,1522	0,4001

Accuracy	1	2	3	4	5	6	7	8	9	10	Avg	StdDev	Min	Max
Accuracy	0,8274	0,8264	0,8260	0,8261	0,8278	0,8278	0,8278	0,8262	0,8253	0,8266	0,8268	0,0009	0,8253	0,8278
AUC	0,6243	0,6231	0,6216	0,6189	0,6255	0,6240	0,6208	0,6148	0,6157	0,6207	0,6209	0,0036	0,6148	0,6255
F1	0,8993	0,8986	0,8985	0,8990	0,8994	0,8995	0,8998	0,8991	0,8984	0,8990	0,8991	0,0005	0,8984	0,8998
MCC	0,4217	0,4188	0,4150	0,4183	0,4257	0,4252	0,4264	0,4212	0,4162	0,4193	0,4208	0,0040	0,4150	0,4264
Accuracy + Diversity (25%)	1	2	3	4	5	6	7	8	9	10	Avg	StdDev	Min	Max
Accuracy	0,8235	0,8182	0,7959	0,7733	0,7538	0,8186	0,7954	0,8182	0,7873	0,7780	0,7962	0,0234	0,7538	0,8235
AUC	0,6102	0,6199	0,6320	0,6124	0,5669	0,6221	0,6373	0,6281	0,6029	0,6170	0,6149	0,0198	0,5669	0,6373
F1	0,8976	0,8931	0,8755	0,8609	0,8504	0,8932	0,8739	0,8925	0,8723	0,8640	0,8773	0,0162	0,8504	0,8976
MCC	0,4073	0,3804	0,3365	0,2636	0,1702	0,3805	0,3487	0,3866	0,2770	0,2777	0,3229	0,0743	0,1702	0,4073
Accuracy + Diversity (50%)	1	2	3	4	5	6	7	8	9	10	Avg	StdDev	Min	Max
Accuracy	0,7948	0,7958	0,7267	0,6820	0,8078	0,8233	0,8194	0,8241	0,8209	0,8255	0,7920	0,0487	0,6820	0,8255
AUC	0,6149	0,6297	0,5736	0,6089	0,6076	0,6116	0,6304	0,6112	0,6173	0,6127	0,6118	0,0156	0,5736	0,6304
F1	0,8764	0,8758	0,8282	0,7795	0,8868	0,8973	0,8931	0,8979	0,8952	0,8987	0,8729	0,0391	0,7795	0,8987
MCC	0,3153	0,3303	0,1542	0,2047	0,3345	0,4048	0,3872	0,4104	0,3944	0,4184	0,3354	0,0906	0,1542	0,4184
Accuracy + Diversity (75%)	1	2	3	4	5	6	7	8	9	10	Avg	StdDev	Min	Max
Accuracy	0,6827	0,8233	0,8247	0,8243	0,7588	0,7200	0,7200	0,8249	0,7845	0,7761	0,7739	0,0523	0,6827	0,8249
AUC	0,5569	0,6132	0,6122	0,6110	0,5936	0,5916	0,5909	0,6111	0,5365	0,6132	0,5930	0,0266	0,5365	0,6132
F1	0,7619	0,8972	0,8982	0,8981	0,8516	0,8172	0,8201	0,8985	0,8765	0,8629	0,8582	0,0461	0,7619	0,8985
MCC	0,1400	0,4036	0,4137	0,4121	0,2200	0,1916	0,1835	0,4148	0,1748	0,2694	0,2823	0,1156	0,1400	0,4148

F1	1	2	3	4	5	6	7	8	9	10	Avg	StdDev	Min	Max
Accuracy	0,8278	0,8262	0,8264	0,8255	0,8264	0,8247	0,8272	0,8272	0,8257	0,8262	0,8263	0,0009	0,8247	0,8278
AUC	0,6243	0,6135	0,6155	0,6121	0,6133	0,6125	0,6179	0,6195	0,6141	0,6182	0,6161	0,0039	0,6121	0,6243
F1	0,8995	0,8992	0,8992	0,8988	0,8994	0,8982	0,8996	0,8995	0,8988	0,8989	0,8991	0,0004	0,8982	0,8996
MCC	0,4252	0,4231	0,4222	0,4185	0,4243	0,4138	0,4257	0,4240	0,4194	0,4187	0,4215	0,0038	0,4138	0,4257
F1 + Diversity (25%)	1	2	3	4	5	6	7	8	9	10	Avg	StdDev	Min	Max
Accuracy	0,8209	0,8227	0,8241	0,8231	0,7778	0,8239	0,8237	0,7698	0,8237	0,8231	0,8133	0,0209	0,7698	0,8241
AUC	0,6286	0,6100	0,6109	0,6096	0,6185	0,6117	0,6106	0,5994	0,6116	0,6090	0,6120	0,0074	0,5994	0,6286
F1	0,8944	0,8970	0,8979	0,8973	0,8637	0,8977	0,8977	0,8591	0,8976	0,8974	0,8900	0,0151	0,8591	0,8979
MCC	0,3933	0,4035	0,4097	0,4052	0,2794	0,4091	0,4083	0,2483	0,4078	0,4061	0,3771	0,0603	0,2483	0,4097
F1 + Diversity (50%)	1	2	3	4	5	6	7	8	9	10	Avg	StdDev	Min	Max
Accuracy	0,8211	0,8249	0,7771	0,7009	0,8230	0,8239	0,7845	0,8251	0,8235	0,8219	0,8026	0,0399	0,7009	0,8251
AUC	0,6240	0,6104	0,6139	0,5777	0,6220	0,6114	0,5352	0,6121	0,6306	0,6073	0,6045	0,0281	0,5352	0,6306
F1	0,8949	0,8985	0,8636	0,8010	0,8945	0,8978	0,8766	0,8985	0,8960	0,8967	0,8818	0,0307	0,8010	0,8985
MCC	0,3928	0,4162	0,2714	0,1643	0,3915	0,4085	0,1740	0,4162	0,4059	0,4001	0,3441	0,1015	0,1643	0,4162
F1 + Diversity (75%)	1	2	3	4	5	6	7	8	9	10	Avg	StdDev	Min	Max
Accuracy	0,7182	0,7570	0,7570	0,8261	0,8209	0,7556	0,8213	0,7306	0,8198	0,8231	0,7830	0,0432	0,7182	0,8261
AUC	0,5923	0,5928	0,5934	0,6153	0,6217	0,5900	0,6235	0,5749	0,6301	0,6106	0,6045	0,0181	0,5749	0,6301
F1	0,8164	0,8503	0,8503	0,8990	0,8949	0,8496	0,8951	0,8298	0,8934	0,8973	0,8676	0,0316	0,8164	0,8990
MCC	0,1888	0,2156	0,2170	0,4206	0,3928	0,2101	0,3945	0,1598	0,3891	0,4042	0,2993	0,1080	0,1598	0,4206

MCC	1	2	3	4	5	6	7	8	9	10	Avg	StdDev	Min	Max
Accuracy	0,8278	0,8263	0,8269	0,8255	0,8274	0,8257	0,8276	0,8243	0,8274	0,8268	0,8266	0,0011	0,8243	0,8278
AUC	0,6208	0,6132	0,6152	0,6118	0,6400	0,6185	0,6219	0,6101	0,6234	0,6142	0,6189	0,0087	0,6101	0,6400
F1	0,8998	0,8992	0,8995	0,8988	0,8980	0,8984	0,8995	0,8981	0,8993	0,8996	0,8990	0,0006	0,8980	0,8998
MCC	0,4258	0,4231	0,4254	0,4193	0,4222	0,4153	0,4248	0,4125	0,4231	0,4264	0,4218	0,0047	0,4125	0,4264
MCC + Diversity (25%)	1	2	3	4	5	6	7	8	9	10	Avg	StdDev	Min	Max
Accuracy	0,8257	0,8239	0,8247	0,8203	0,8009	0,8078	0,8243	0,8227	0,8270	0,8251	0,8202	0,0087	0,8009	0,8270
AUC	0,6153	0,6198	0,6241	0,6273	0,6483	0,6390	0,6223	0,6106	0,6187	0,6146	0,6240	0,0116	0,6106	0,6483
F1	0,8987	0,8971	0,8973	0,8940	0,8776	0,8828	0,8972	0,8970	0,8994	0,8983	0,8939	0,0075	0,8776	0,8994
MCC	0,4184	0,4078	0,4090	0,3919	0,3673	0,3889	0,4078	0,4027	0,4235	0,4137	0,4031	0,0165	0,3673	0,4235
MCC + Diversity (50%)	1	2	3	4	5	6	7	8	9	10	Avg	StdDev	Min	Max
Accuracy	0,8235	0,8199	0,7452	0,8203	0,8202	0,7989	0,8237	0,8235	0,8243	0,8251	0,8125	0,0248	0,7452	0,8251
AUC	0,6187	0,6242	0,5742	0,6273	0,6278	0,6129	0,6103	0,6114	0,6119	0,6121	0,6131	0,0153	0,5742	0,6278
F1	0,8969	0,8940	0,8405	0,8940	0,8939	0,8498	0,8977	0,8975	0,8980	0,8986	0,8861	0,0217	0,8405	0,8986
MCC	0,4039	0,3919	0,1761	0,3919	0,3915	0,3181	0,4081	0,4067	0,4107	0,4128	0,3712	0,0739	0,1761	0,4128
MCC + Diversity (75%)	1	2	3	4	5	6	7	8	9	10	Avg	StdDev	Min	Max
Accuracy	0,8207	0,8243	0,7912	0,6761	0,7442	0,8255	0,8196	0,7165	0,8229	0,8235	0,7864	0,0546	0,6761	0,8255
AUC	0,6269	0,6123	0,6079	0,5805	0,5836	0,6146	0,6029	0,5858	0,6104	0,6092	0,6034	0,0152	0,5805	0,6269
F1	0,8943	0,8980	0,8747	0,7657	0,8412	0,8986	0,8954	0,8172	0,8971	0,8976	0,8680	0,0457	0,7657	0,8986
MCC	0,3942	0,4209	0,2923	0,1716	0,1871	0,4166	0,3875	0,1719	0,4049	0,4084	0,3255	0,1089	0,1716	0,4209