Facultat d'Informàtica de Barcelona (FIB)

universitat politècnica de catalunya (upc) – BarcelonaTech

Master thesis

Learning with a neural network based on similarity measures

Master in Innovation and Research in Informatics

DATA SCIENCE

Author:
Martí Cardoso i Sabé

Advisor:
Lluís A. Belanche Muñoz
Computer Science Department

29th January 2020

Abstract

Currently, in machine learning, there is a growing interest in finding new and better predictive models that can deal with heterogeneous data and missing values. In this thesis, two learning algorithms are proposed that can deal with both issues.

The first learning algorithm that is studied consists of a neural network based on similarity measures, the Similarity Neural Network (SNN). It is a two-layer network, where the hidden layer computes the similarity between the input data and a set of prototypes, and the output layer gathers these results and predicts the output. In this thesis, several variants of this algorithm are proposed and it is analyzed which one performs better. Some of these variants are the way to choose the prototypes or how to set the parameters of the activation function. A full analysis is performed in the experiments section.

Secondly, an Ensemble of SNNs is also proposed. The purpose of using an ensemble is to increase predictive performance, reduce variability and reduce learning time complexity. This second learning algorithm combines the predictions of a set of SNNs and gives the response of the ensemble based on these predictions. For this algorithm, several ensemble learners are proposed (in other words, different ways to combined these predictions). These variants are analyzed with a set of experiments.

The main goal of this thesis is to understand these two methods, derive training algorithms and compare them with traditional learning algorithms, such as the classical Random Forest. The results of the experiments show a competitive performance of both methods, obtaining similar results than the Random Forest and improving it in some problems. 16 datasets with heterogeneous data and missing values are tested, some of them large and difficult problems. About the SNN, with these experiments, it is found that adding regularization to the network has a high influence on the model. About the ensemble, the experiment results suggest that the simplest ensemble learner (mean or majority vote of the SNNs) is the one that performs better. Among the two proposals, both get similar and quite good performance metrics but the ensemble obtains slightly better predictions.

Acknowledgements

First of all, I would like to express my sincere thanks of gratitude to Lluís Belanche for being my master thesis advisor. I am very grateful for his guidance, advice, suggestions, warnings and support that he has given me during the last six months when I have been developing this thesis.

Secondly, I would also like to thanks to all my family for their support during this time. Especially to my parents and sister for their help and for being patients with me. And finally, to Júlia for her love, understanding and continuous support.

Many thanks to all of you!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, it is explained what this thesis is about, the main objective and motivations of developing it and the general structure of this document.

1.1 Context

Nowadays, machine learning has become very popular and it is a trending topic of research in the scientific community. Many researchers are working on this field, but not only in the research sector also in the industry, where the companies have a focus on this area because they found that these techniques can be used in real-world problems.

Supervised learning is one of the most popular machine learning tasks. It consists on given a dataset (a set of observations with P features and its responses), it trains a model that predicts the response in a generalized way to unseen situations.

In this group, it exists a method that is not very popular and has a ground for research: *similarity-based learning techniques* applied to heterogeneous data. As the name itself suggests, this method uses the similarity between observations instead of features/attributes to train the model (as most of the methods do). So, this technique learns and discovers knowledge based on similarities between observations.

In this thesis, it is studied a particular case of *similarity-based learning method*: Similarity Neural Network (SNN). It is a new learning algorithm that consists of a neural network where its neurons are based on similarity measures. As an overview, it is a two-layer network, where the hidden layer computes the similarity between the input data and a set of prototypes, and the output layer gathers these results and predicts the output variable.

Some of the strong points and advantages of the SNNs are:

- Heterogeneous data.

  An SNN network can deal with problems with heterogeneous data (different types of variables). Each variable of the dataset has its own similarity measure, the one that betters fits it. Of course, whenever available, an expert should choose the best similarity functions for each variable.

- Missing data.

  As it is explained in future sections, the similarity between observations allows features to be missing, so these learners can deal with missing values.

- Interpretability

  There is an increase of interpretability compared to classical neural networks: the hidden neurons compute the similarity between the input and a set of prototypes (a known set of observations), and the response of the network is the result of a linear combination of these similarities. So, the predictions of the network can be better interpreted.

These are some of the main features of the SNNs, and they are some of the reasons why it makes sense to research in this area. So, this thesis is focused on the SNNs, trying to understand how they work, how they learn patterns in the data and it continues the current research on this learning method.

1.2 Objectives

This thesis has the main goal of understanding the Similarity Neural Network (SNN), derive training algorithms, analyse the impact of adding the similarity layer and compare it with traditional learning algorithms, such as Random Forest. Also, an Ensemble of SNNs is proposed, so for this second learning algorithm, it should be done the same tasks (derive training algorithms, compare with other methods, ...).

This goal is very general and in order to accomplish it, it has been divided into several concrete tasks/objectives:

1. Understand and design the Similarity Neural Network (SNN).
2. It does not exist a complete implementation of an SNN. Only a few researchers have implemented some prototypes. In this thesis, it is desired to give an implementation that could be used for any problem/dataset (and any data scientist can easily use it).
3. Add regularization to the Similarity Neural Network.
4. Research and propose a method to optimize the p hyper-parameter of the activation function of the hidden neurons (more details in future sections).
5. Design an Ensemble of Similarity Neural Networks.
6. Implement the Ensemble of Similarity Neural Networks.
7. Test the performance of the SNN and Ensemble of SNNs by designing experiments.
8. Make a comparison between the Similarity Neural Network and the methods in the literature.
9. Test the performance of the learning methods with large problems (more than 50 000 observations).

1.3 Document structure

My master thesis is organized in the following way:

- **Chapter 1: Introduction**. In the first chapter, the problem that we are going to study and the main objectives of this thesis are explained.
- **Chapter 2 Related Work and State-of-the-Art**. It is explained the current literature of the similarity-based learners for heterogeneous data, in particular, the SNN.
- **Chapter 3: Preliminaries**. In the third chapter, some of the preliminaries concepts used in this thesis are explained. They should be understood in order to fully understand this document.
- **Chapter 4: The proposals**. It is presented the design of the SNN and the Ensemble of SNNs.
- **Chapter 5: Implementation and Use**. It is given an overview of how it was implemented and how to use the SNNs and Ensemble of SNNs.
- **Chapter 6: Experiments**. In this chapter, the experiments designed to test the SNNs and Ensemble of SNNs are explained. For each experiment, it is described the reasons for designing it and the main factors that are analyzed.

- **Chapter 7: Results**. In the results chapter, the numerical and graphical results obtained with the experiments are presented and explained.
- **Chapter 8: Conclusions**. It is explained the main conclusions of this thesis.
- **Chapter 9: Future work**. Finally, there are some of the future works that could be done after this thesis.

Chapter 2

Related Work and State-of-the-Art

In this chapter, it is briefly explained the current research on *similarity-based methods* for heterogeneous data and the need for their existence in supervised learning.

The similarity measure is not a new research field in machine learning, quite the opposite. For example, it has been used for years for unsupervised learning tasks like clustering, and the use of similarity measures is quite popular and gives good results in this field.

In the supervised learning world, the use of these similarity measures is much less explored (less researched and used). Some of the advantages of using similarities are that it allows the learner to deal with heterogeneous data and missing values. Quite the opposite than most of the classical supervised learning techniques that are feature-based algorithms (e.g. MLP-NN) or kernel-based algorithms (e.g. SVM), and they cannot deal with heterogeneous data and missing values (a pre-processing step must be done). Very few researches tried to introduce heterogeneous data and similarities in these supervised learning algorithms (later, some of these researches are explained).

In this thesis, we are going to focus on the **similarity-based techniques** applied to heterogeneous data, in particular, the Similarity Neural Network (SNN). As said before, this network natively deals with heterogeneous data and missing values. These are some of the weak points of most of the traditional learners, so here is where the similarity-based techniques start to take importance.

Let us review how traditional learners deal with these issues:

- **How do learners deal with missing values?**

  Most of the learning algorithms cannot deal with missing values and they fail. One, as a data scientist, must have to manage to remove or impute these missing values. Some of the most common solutions are:

  - Ignore all observations with missing values.
  - Replace all missing values with column mean.
  - Replace all missing values with some constant (e.g. -999,-1).
  - Impute missing values with other methods.
  - ...

  These methods have to be executed before the learning algorithm, so if the dataset of the problem has missing values, one must have to spend more time in the pre-processing stage. Also, the imputation method choice can be a key fact for the model, choosing one or another can add bias and variance to the data or it can loss relevant information.

  In the other hand, a few learners can deal with missing values (e.g. decision trees), and thus, there is no need for imputation. In this second group is where the similarity-based learners belong. As the similarity measures are between observations instead of features, when the dataset has missing values, it does not take into account the missing features of the observations when the similarity is computed.

- **How do learners deal with heterogeneous data?**

  Some learners can deal with only numerical variables, then for these methods, it is usually mapped the non-numerical variables to numerical values. For example, a categorical variable could be mapped into m new binary variables (one for each category) or it could be mapped to a value between 1 and m. Also, some other learners can only deal with categorical variables, then the numerical variables are discretized into intervals and new categorical variables are created. Despite this, changing the type of a variable (e.g. transforming numerical variables to categorical and vice-versa) is not the most adequate transformation to do and the treatment of these variables could be improved using different types of similarity measures.

  So, the similarity-based methods treat better the different types of variables because one can define a more specific measure to be used for each variable (the one that fits better): numeric, nominal, ordinal, cyclic,... . Doing that, we can compare the features in a better way and get more relevant information about these variables.

Summarizing, some of the weak points of the most used learning algorithms are that they cannot deal with missing values and the treatment that they do to heterogeneous data. These are some of the strong points of the similarity-based techniques.

Finally, the current state of the art of similarity-based learners for heterogeneous data is reviewed:

Similarity-based learners for heterogeneous data

As said before, the similarity-based techniques applied to heterogeneous data is a field that needs more research, the research on other techniques is much more extended than in similarity-based learners. But the few research on this area show some interesting results and that the similarity-based methods improve some of the classical learning methods. I am going to explain briefly some of these researches.

In [1], [2] and [3], it is proposed a two-layer neural network where the first layer computes a user-defined similarity function between observations and a reduced set of them (called centres or prototypes). The results in [1] show a competitive or better generalization performance than that found in the literature and in [2] it is made a comparison between the network and an RBF network for the Horse Colic problem. [4] compares dissimilarity-based classifiers with traditional feature-based classifiers (e.g. linear and nonlinear SVMs) and as averaged over more than 300 datasets it performs similar or better than the feature-based classifiers.

In general, all research follows the design of [1], where the neural network is a two-layer network where the first layer computes a user-defined similarity function between observations and a set of prototypes, and the second layer gathers all these similarities and gives the response of the network. This learning algorithm is called **Similarity Neural Network** (SNN) and it is the case of study for this project (in section 4.1 SNNs are explained in full details). There is much remaining work to be done until SNNs can be a viable off-the-shelf modelling method.

Chapter 3

Preliminaries

In this section, some of the theoretical concepts related to this thesis are briefly explained. These concepts are very important and should be understood in order to understand the full description of this document. Firstly, the concept of similarity measures is explained, a key concept on which this thesis is based on. Secondly, it is briefly introduced the Artificial Neural Networks (ANN), the ensemble methods and the Generalized Linear Model (GLM). And finally, the gradient descent algorithm used in this thesis to optimize the parameters of the models and some other machine learning concepts are explained.

3.1 Similarity measures

A similarity measure [6] is a function that compares two objects and expresses how similar or likely these two objects are. Let s be the similarity function, and X a non-empty space where the objects belong, then s is defined as:

$$s : X \times X \to I_s \subset \mathbb{R}$$

and s is assumed to be upper bounded, exhaustive and total function. Therefore, I_s is upper bounded and $\sup_{\mathbb{R}} I_s$ exists.

In a general sense, the measure gives a numerical value that indicates the degree of coincidence between two objects. So, as more similar or equal two objects are, higher this value should be, and the other way around, as more different they are, this value should be lower.

First, we are going to start defining similarities between two objects (objects with more than one feature), and then it is explained how to compute the similarity between basic data types. In this thesis, we are going to use similarity measures defined in the common codomain $I_s = [0, 1]$.

3.1.1 Gower's similarity measure

The Gower's general similarity measure [7] is a coefficient that computes how similar two observations are. In this case, each observation is a heterogeneous set of variables. The Gower's similarity coefficient can be computed with the following formula:

$$s_g(x_i, x_j) = \frac{\sum_{k=1}^{P} s_k(x_{ik}, x_{jk}) \cdot w_k \cdot \delta_{ijk}}{\sum_{k=1}^{P} w_k \cdot \delta_{ijk}}$$

where P is the total number of variables, $s_k(x_{ik}, x_{jk})$ is the similarity between observation i and j for the feature k, w_k is the weight of the feature k and δ_{ijk} is the possibility of making the comparison between the two observations for the feature k (it will be 1 if they can be compared, 0 otherwise).

In other words, this coefficient is a weighted mean of the similarities of the variables (of course, without taking into account the variables that cannot be compared).

Two features cannot be compared, and thus δ_{ijk} is 0, when one of the two values (x_{ik} or x_{jk}) is missing, and usually they can be compared when both are not missing ($\delta_{ijk} = 1$) .

3.1.2 Data type similarity measures

The type of a variable determines the similarity function to be used for this variable. Here, the most common data type similarity measures are presented (most of them were proposed by Gower [7]):

- **Continuous variable**. The similarity between two numerical values of a variable x is defined by its absolute difference and the range where this variable belongs:

$$s(x_i, x_j) = 1 - \frac{|x_i - x_j|}{R}$$

  where x_i and x_j are the two values to compare and R is the range of the variable (for this thesis, I take R as the range in the sample).

- **Nominal variable.** The nominal variables do not have an order between the different modalities of the variable, so the similarity can only check for equality. This similarity measure is:

$$s(x_i, x_j) = \begin{cases} 1 & \text{if } x_i = x_j \\ 0 & \text{if } x_i \neq x_j \end{cases}$$

- **Ordinal variable**. An ordinal variable is similar to a nominal variable, but it exists a clear order in the modalities of the variable. Supposing that the modalities are codified from 1 to k (following the right order of the variable), the similarity is computed as:

$$s(x_i, x_j) = 1 - \frac{|x_i - x_j|}{k - 1}$$

- **Symmetric binary variable**. A binary variable is symmetric when both of its states (absent and present) have the same weights. So, in this case, two absent are similar. The similarity for symmetric binary variables is:

$$s(x_i, x_j) = \begin{cases} 1 & \text{if } x_i \text{ and } x_j \text{ are present or } x_i \text{ and } x_j \text{ are absent} \\ 0 & \text{otherwise} \end{cases}$$

- **Asymmetric binary variable**. A binary variable is asymmetric when its states have different weights, and thus two absent are not similar. The similarity for asymmetric binary variables is:

$$s(x_i, x_j) = \begin{cases} 1 & \text{if } x_i \text{ is present and } x_j \text{ is present} \\ 0 & \text{otherwise} \end{cases}$$

  For asymmetric binary variables, $\delta_{ijk} = 0$ when both values are in the absent case (and also when one of the two is missing).

These are the most common similarity measures for basic data types. There exist other types that were not explained (e.g. fuzzy and cyclic variables) because they are not used in this thesis and the R function that implements the Gower's similarity coefficient does not support them.

3.2 Artificial Neural network (ANN)

An Artificial Neural Network is a statistical model that takes its inspiration from the human brain and in recent years its popularity has increased a lot. The main idea behind ANNs is that if we can understand how the brain works and solves a given problem, we can define solutions to these tasks as formal algorithms and implement them on computers.

An ANN consists of a collection of neurons and the connections between them. Each neuron receives information from other neurons and propagates new information to the forward ones. So, an ANN can be seen as a directed and weighted graph.

One of the most used ANNs is the Multilayer Perceptron Neural Network (MLP-NN) [8], it is a feed-forward ANN (unidirectional) with one input layer, one or more hidden layers and the output layer. The input layer contains the input variables, each variable is connected with the first hidden layer neurons, the first hidden layer neurons are connected with all second hidden layer neurons, until the last hidden layer that its neurons are connected with the output layer neurons (response of the MLP-NN). A hidden neuron receives a set of inputs and generates an output. In this case, its output is the result of applying an activation function to the weighted sum of its inner connections (the output of a neuron is computed as follows: $o(x) = f(w^t x)$ where f is the activation function, w the neuron weights and x the values of the inner connections). Training the network consists of setting the weight of each connection (it is usually used the back-propagation algorithm). Figure 3.1 shows an example of this type of network.

The Radial Basis Function Neural Network (RBF-NN) [9] is another type of ANN with one hidden layer that uses the radial basis functions as activation functions. Typically, the Euclidean distance and the Gaussian distribution are used, and thus, the output of the hidden neurons is: $o(x) = exp(-\beta||x - w||^2)$ where w is the neuron weights and x the input values. This network is very similar to the MLP-NN, the main differences are the activation function and that MLP-NN computes the output of the neurons by the dot product between inputs and weights, and RBF-NN computes the euclidean distance between them (so now these weights can be interpreted as prototypes).

Finally, the Similarity Neural Network (SNN), that it is proposed and studied in this thesis, is similar to the previous networks, but for computing the output of the neurons, it is used the similarity between a selected prototype and the inputs. We are going to explain deeply this network in future sections.

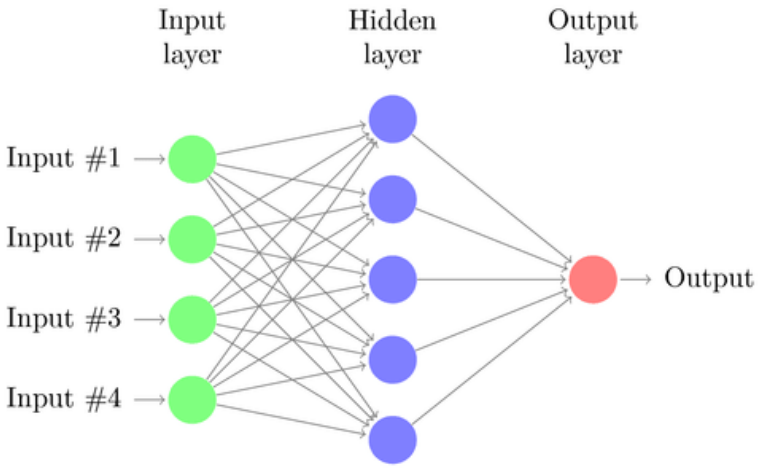

Figure 3.1: Example of a feed-forward neural network.

3.3 Ensemble methods

In machine learning, ensemble methods are a type of supervised learning algorithms that combine multiple learners into one predictive model. The main idea behind these methods is that the combination of these learners obtains better predictive performance than each one of the learners alone. Some of the most used ensemble methods are Bagging, Boosting and Stacking.

For example, the Bagging method [10] creates a set of learners of the same type on which each one is trained with a different subset of instances that were sampled with replacement from the training dataset. Then the prediction of the bagging is the average over the learners when a numerical variable is predicted, and when predicting a class it does a plurality vote over the learners. It is usually used with decision trees, but of course, any other type of learner can be used.

3.4 Generalized linear models

A Generalized Linear Model (GLM) [11] is a statistical model that generalizes the classical linear regression in the way that it does not assume that the response variable has the errors normally distributed. Therefore, GLMs are models in which the response variable density is in the exponential family (e.g. Poisson, Gaussian, Bernoulli, Gamma, ...) and where the mean parameters are a linear combination of the explanatory variables, passed through a possibly nonlinear function, such as the logistic function. We can define the conditional distribution of the response variable Y given the explanatory variables X as:

$$\mathbb{E}(Y|X) = h^{-1}(\beta^T X + \beta_0)$$

where h is the link function (and h^{-1} its inverse) and β the unknown coefficients used to do the linear combination with the explanatory variables.

In this case, one should decide the suitable distribution for the response variable, and thus its convenient link function. For example, the Gaussian distribution uses the identity as the link function (then, the problem becomes the classical linear regression), or for the Bernoulli distribution, the logit function is used (logistic regression).

When training the model, this generality of the GLM comes at a cost because, in general, we need an iterative procedure for the optimization of βs. A popular procedure is to set it up as a maximum likelihood problem and use a numerical optimization method (e.g. Newton-Raphson). For the Gaussian distribution case (classical linear regression), the solution can be found analytically and there is no need for an iterative procedure. And for the logistic regression, it is typically used the Iterated Reweighted Least Squares (IRLS) to optimize βs.

3.5 Gradient descent method

In this section, the unconstrained optimization algorithms [12] of a given function (called f) are briefly explained. In a general sense, these algorithms start with an initial solution and it is iteratively updated/improved until a certain stopping criterion is satisfied.

The algorithm updates the current solution as follows:

$$x^{k+1} = x^k + \alpha^k d^k$$

where x is the solution, k the current iteration, d^k is the descent direction and α^k the step length in the current iteration. In each iteration, we have to find a direction and a step length to update the solution.

For the descent direction, there are several methods, the simplest one is the gradient descent, a first-order method that sets the descent direction as the negative of the gradient ($d^k = -\nabla f(x^k)$). Other methods to find the descent direction are the Conjugate Gradient, Quasi-Newton and Newton method (second-order method).

For the step length α there are two main methods: exact line search (finds the optimum value of α for a given direction) or backtracking line search (an iterative procedure to find an α that

guarantees the given acceptability criterion). Usually, the second method is used because the exact line search is more costly.

So, in each iteration, it is chosen a descent direction and a step length, and the solution is iteratively updated until a certain criterion is satisfied. In order to use this optimization algorithm we must provide the function to optimize and its first derivative (and also the second derivative for second-order methods). In machine learning and also in this thesis, this technique is used to optimize the parameters of the models.

3.6 Cross-validation

The Cross-validation methods are a family of model validation techniques that give an estimation of how a statistical analysis will generalize to unseen data.

The *hold-out* method is the simplest case and it consists of splitting the data into two parts: one for training the model and the other for testing it. A common split is to use $2/3$ of the data for training and the remaining $1/3$ for testing.

Secondly, the k-fold cross-validation consists of dividing randomly the data into k parts of equal size. For each of these parts, we keep it out from the training set and it is used all others $k-1$ parts to fit the model, once the model is fitted we use the remaining part to check the goodness of fit of the model. It is repeated k times, one for each part. This method is an improvement over the holdout method, but it needs to fit k models.

A specific case of k-fold cross-validation is the Leave-One-Out Cross-Validation (LOO-CV), where k is the total number of observations. So we remove one observation from the data, we fit the model and it is predicted and tested this observation.

3.7 Performance metrics

The performance metrics are used to test the models and compare them. It is a measure of goodness of fit of a model. In this document, it is used the **Normalized Root Mean Square Error (NRMSE)** for regression problems and **Accuracy** for classification.

Given a dataset with N observations, the accuracy is defined as:

$$Accuracy = \frac{\sum_{i=1}^{N} \mathbb{1}\{y_i = \hat{y}_i\}}{N}$$

where y_i is the real response value of the ith observation, $\hat{y}_i$ is the prediction for the ith observation and $\mathbb{1}$ is a function that gives 1 when both values are equal and 0 otherwise.

And the NRMSE error measure is:

$$NRMSE = \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2}$$

Usually, these performance metrics are used to test the goodness of fit of the model, so it is computed with the test set.

Chapter 4

The proposals

In this chapter, the two main proposals studied in this thesis are explained:

- SNN
- Ensemble of SNNs

Both are machine learning algorithms that solve supervised problems for heterogeneous data (datasets with N observations where each observation has P features and a response value to predict). The first proposal consists of a neural network based on similarities and the second is an ensemble of the first proposal. Let us explain them in details.

4.1 Similarity Neural Network (SNN)

As an overview, the Similarity Neural Network (SNN) is a two-layer neural network where the hidden layer computes the similarity between the input data and a set of prototypes and the output layer gathers these results and predicts the response.

Firstly, it is explained the design and the main framework of an SNN, and secondly, it is explained the way to train it.

4.1.1 General framework

As said before, the SNN is a neural network where firstly it is computed the similarity between the input space and a set of prototypes and, secondly, given these similarities it is predicted the response value.

The SNN receives an input x with p features (x_i is the feature i of input x) and the goal is to predict the response variable y of this input. The network is structured in two layers: the hidden layer and the output layer.

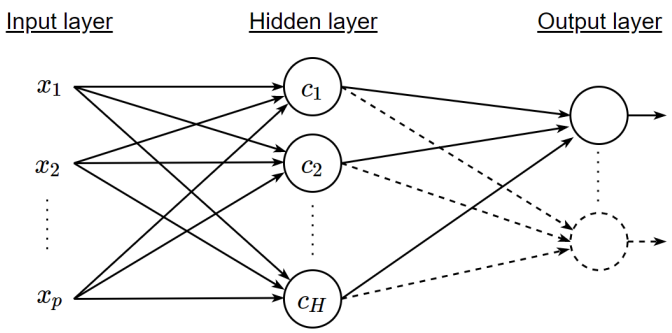

Figure 4.1: SNN network example.

In the hidden layer, it is computed the Gower's similarity between the input and a set of H prototypes chosen in the learning stage (we will explain how to chose them in future sections), each prototype corresponds to a neuron in the hidden layer, so this layer has H neurons. And the outputs of the hidden neurons are the results of applying an activation function to these similarities. As the Gower's similarities go from $[0, 1]$, we apply a sigmoid-like function that given a value in the interval $[0, 1]$ it returns a value in $[0, 1]$. This activation function is called f_p and it is deeply explained below.

Finally, the output layer gathers all outputs of the hidden neurons and builds a Generalized Linear Model (GLM) that predicts the response variable.

At this point, it was given an overview of the SNN framework, Figure 4.1 shows a graphical representation of the network. Now, the two layers of the network are explained in more details:

4.1.1.1 Hidden layer

The hidden layer is a set of H hidden neurons where each one of them has assigned one prototype (called c_i). Then, the output of the hidden neuron i is the result of applying the activation function to the similarity of Gower between the input and the prototype of the hidden neuron (c_i). So, the output of the hidden neurons is computed as follows:

$$h_i(x) = f_p(s_g(x, c_i)) \qquad \text{for } i \text{ in } 1..H$$

where f_p is the activation function, s_g the Gower's similarity and c_i is the prototype corresponding to the ith neuron.

- **Gower's similarity**.

  The Gower's similarity between two observations can be computed with the following formula:

$$s_g(x_i, x_j) = \frac{\sum_{k=1}^{P} s_k(x_{ik}, x_{jk}) \cdot w_k \cdot \delta_{ijk}}{\sum_{k=1}^{P} w_k \cdot \delta_{ijk}}$$

  where P is the number of variables, $s_k(x_{ik}, x_{jk})$ is the similarity between observation i and j for the feature k, w_k is the weight of the feature [1] and δ_{ijk} is the possibility of making the comparison between the two observations for the feature k. More details in Section 3.1.1.

- **Activation function**

  The activation function f_p used in the hidden neurons is defined by [13] and it is a sigmoid-like automorphism (a monotonic bijection) in $[0, 1]$. This function adds a nonlinear component to the network and it is a function that maps values from the interval $[0, 1]$ into $[0, 1]$.

  The activation function f_p is defined as:

$$f_p(\cdot) = f(\cdot, p)$$

  And:

$$f(x, p) = \begin{cases} \frac{-p}{(x-0.5)-a(p)} - a(p) & \text{if } x \leq 0.5 \\ \frac{-p}{(x-0.5)+a(p)} + a(p) + 1 & \text{if } x \geq 0.5 \end{cases}$$

  where $p > 0 \in \mathbb{R}$ is a parameter controlling the shape. f_p is a family of functions that are all continuous bijections in $[0, 1]$, fulfilling $\forall p \in \mathbb{R}+,\ f(0, p) = 0,\ f(1, p) = 1,\ \lim_{p\to\infty} f(x, p) = x$ and $f(x, 0) = H(x - 0.5)$, being H the Heaviside function. And $a(p)$ is the solution of the equation $a(p)^2 + \frac{a(p)}{2} - p = 0$, obtained by imposing the first two equalities. Figure 4.2 (Right) shows a graphical representation of the behavior of the function for several values of p. More details about the activation function in [13].

[1] For the experiments in this thesis, I have assumed that the features are equally weighted, so w_k is 1 for all k.

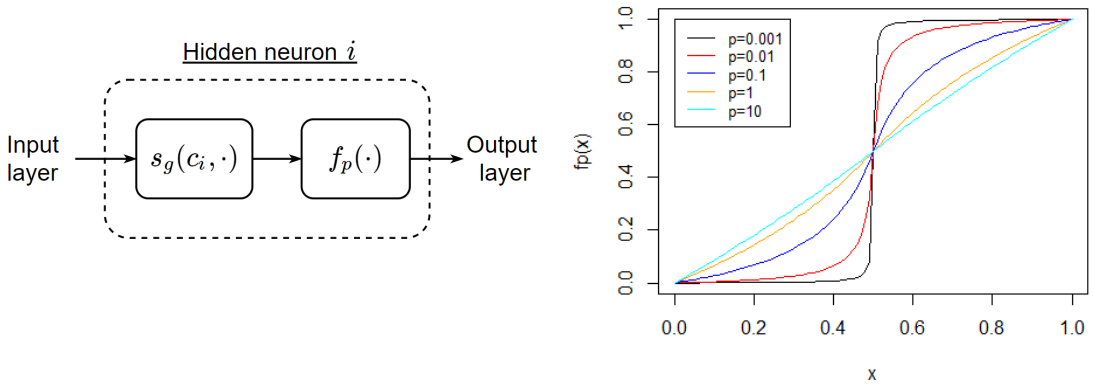

Figure 4.2: (Left) Hidden neuron framework (Right) Activation function f_p for different values of p

4.1.1.2 Output layer

The output layer is a Generalized Linear Model (GLM), so it creates a linear model that receives the similarities computed in the hidden layer and predicts the response. The type of response variable determines the activation function of the neurons in this layer. The SNN can deal with these three types of response variables:

- **Numerical**. In this case, we have a regression framework and the activation function used is the identity. This layer has only one output neuron and the prediction of the SNN is :

$$\hat{y}(x) = \sum_{j=1}^{H} \alpha_j \cdot h_j(x)$$

  where α_j are the weights of the linear model (these weights were computed at the learning stage).

- **Binary/Binomial**. When the variable is binary we have the logistic regression framework, so the activation function is the *logistic*. Again, there is only one output neuron that has the meaning of the probability of being of response *true*. For this case, the output of the network is:

$$\hat{y}(x) = \frac{1}{1 + exp\left(-\sum_{j=1}^{H} \alpha_j \cdot h_j(x)\right)}$$

- **Multinomial**. When the response variable is multinomial, it is used *softmax* as the activation function. It has an output neuron for each response modality, and the output of each neuron has the meaning of the probability of being of the neuron modality. The prediction of the network is:

$$\hat{y}_k(x) = \frac{exp(\xi(x,k))}{\sum_{k2=1}^{K} exp(\xi(x,k2))} \text{ where } \xi(x,k) = \sum_{j=1}^{H} \alpha_{jk} \cdot h_{jk}(x)$$

  where K is the number of modalities of the response variable and α_{jk} is the weight for the output neuron k and hidden neuron j.

4.1.2 Learning stage

At this point, we have seen the framework of the SNN and how it works. Now, it is explained how to train the network. The three main learning parts are: how to select the prototypes to be used as hidden neurons, how to set the p parameter of the activation function f_p and finally how to train the weights of the output layer (GLM).

4.1.2.1 Prototype selection

The prototype selection is the first thing to be trained in an SNN. It consists on, first, choosing the number of prototypes to be used (in other words, the number of hidden neurons), and secondly select the observations that will be used as prototypes.

Number of prototypes

The number of prototypes can be generated by four different methods, each one uses a different distribution to choose this number. We should be able to define these distributions, so, for this reason, we introduce an hyper-parameter, called hp, that in most of the methods mean the expected proportion of observation that will be selected as prototypes. These are the four methods:

- **Constant**(C). This is the simplest method. The number of prototypes depends only on the size of the problem, and for the same problem, it will always generate the same value. For this case, the hp hyper-parameter means the proportion of observations that will be selected as prototypes. The number of prototypes (H) is:

  $$H = \lfloor hp \cdot N \rfloor$$

  where N is the total number of observations.

- **Uniform** (U). This method uses the uniform distribution to set the number of prototypes. It generates a value between 1 and $hp \cdot N$ using the uniform distribution, so consecutive runs of the method will generate different values. In this case, the hp hyper-parameter means the maximum proportion of observations that could be selected as prototypes. The number of prototypes is:

  $$H = Unif\left(1, hp \cdot N\right)$$

- **Poisson** (P). This method uses the Poisson distribution to set the number of prototypes. We set $\lambda = hp \cdot N$, so the expected value of the distribution is $hp \cdot N$. The number of prototypes is:

  $$H = Poisson\left(\lambda = N \cdot hp\right)$$

- **Binomial** (B). This method uses the binomial distribution to set the number of prototypes. For this distribution, the hp is used as the probability of success on each trial. So, as we have set the number of trials to be the number of observations, this distribution will have an expected value of $hp \cdot N$. The number of prototypes is:

  $$H = Binomial\left(N, hp\right)$$

These are the four methods to generates the number of prototypes, and the user should decide which one should be used. For creating just one SNN, these methods only decide the number of prototypes of the network, but in future sections, it is introduced the Ensemble of SNNs, and what we want for an ensemble is variability among the different learners, so adding this random behaviour on the number of prototypes could be useful for the model.

Selection of prototypes

Once we know the number of prototypes of the network, we have to select which observations are prototypes. Two different methods to do this task are proposed:

- **PAM**. The first method applies the clustering method PAM (Partitioning Around Medoids) [14] to select the prototypes. This clustering technique finds groups of observations that are similar between them and each group has a representative observation (medoid). So, this first method consists of selecting the prototypes as the medoids found with the PAM algorithm. The benefit of this method is that it can select representative observations that are different from each other, but this clustering has a high computational cost.
- **Random**. The second method consists of selecting the prototypes randomly from all observations. It is the opposite of the previous method: it is very fast when selecting the prototypes, but as they are selected randomly, the observations may not be very representative and may be very similar to each other.

In the experiments section, it is analyzed if the use of PAM is useful for predictive tasks, or if it gets similar results than using just random, and thus, we could save computational time using the random method.

4.1.2.2 Value of p (f_p)

Once the prototypes are selected, the next learning phase is to decide the value of p of the activation function f_p. We define four methods to set this value:

Constant

The first and simplest method is to set the p as an hyper-parameter that the user can set. It is predefined as $p = 0.1$ (blue line in Figure 4.2 - Right).

Optimization of p

The second method consists of using optimization techniques to find the best value of p. At this point, the missing parameters to be fitted in the SNN are p and the coefficients/weights of the GLM (αs), but the value of p and α are very related, we cannot optimize one and then the other, we should optimize p and αs at the same time. Optimizing p and then αs could make that the optimization of p is no longer valid (there exists another optimal value for the new αs), and thus, the optimization of p should be repeated, and then αs could have another optimal value that should be optimized, and so on so for. For this reason, p and αs are optimized in the same optimization procedure.

The optimization technique will find a local minimum of the error function using an iterative procedure. In our case, the **gradient descent** method (GD) is used, a first-order optimization algorithm. In order to solve the optimization, we must provide the objective function (in our case the error function) and the first-order derivatives respects to p and αs. The objective function to minimize depends on the type of response variable:

- Regression. For regression problems, we use the Sum of Squares Error (SSE):

$$E(p, \alpha) = \sum_{n=1}^{N} (y_n - \hat{y}(p, \alpha, x_n))^2$$

  where y_n is the real response value of observation n and $\hat{y}(p, \alpha, x_n)$ is the prediction of the SNN with parameters p and α.

- Binary/Binomial. The error function is the cross-entropy for two classes:

$$E(p, \alpha) = -\sum_{n=1}^{N} \Big(y_n ln(\hat{y}(p, \alpha, x_n)) + (1 - y_n) \cdot ln(1 - \hat{y}(p, \alpha, x_n)) \Big)$$

where y_n is 1 when the observation n is positive and 0 if negative and $\hat{y}(p, \alpha, x_n)$ is the prediction (probability of being positive) of the SNN with parameters p and α.

- Multinomial. For multinomial problems it is used the cross-entropy error function for K classes:

$$E(p, \alpha) = -\sum_{n=1}^{N} \left(\sum_{k=1}^{K} y_{nk} \cdot ln(\hat{y}_k(p, \alpha, x_n)) \right)$$

where y_{nk} is 1 when the observation n is of the k modality, 0 otherwise. And $\hat{y}_k(p, \alpha, x_n)$ is the predicted probability of being of k modality for an SNN with parameters p and α.

Appendix A.1 contains all mathematical equations related to these functions and all its derivatives.

When the optimization ends, we have a local minimum for a given p and αs, so we have optimized the value of p at the same time than the αs. I want to stress the fact that when using this method, the regularization of the GLM should not be used because the objective function would not be the one optimized with this procedure.

k-fold Cross-Validation

Another method to choose the value of p is using k-fold cross-validation on the training set. This method tests several values of p, and it selects the p as the one with the best validation error (MSE for regression and accuracy for classification). By default, it does 10-fold cross-validation. I want to stress the fact that for each fold and p, it is fitted the GLM model, so this means that the method has a high computational cost.

Generalized Cross-Validation

Finally, only for regression problems, there exists a way to compute the LOO-CV error fitting the model just once, it is called Generalized Cross-Validation (GCV). This method tests several values of p, and the one with the best GCV error is selected as the best p. It is much faster than the k-fold cross-validation because it fits one model instead of k.

4.1.2.3 GLM

At this point, we know the selected prototypes and the p value to be used in the activation function, so we can compute the outputs of the hidden neurons. The remaining part to be trained is the weights of the GLM (αs). Depending on the response variable it is fitted a different model (types of GLM):

- Regression case: It fits a classical linear model (minimizes the MSE).
- Binomial case: It fits a logistic regression model (minimizes the cross-entropy error function).
- Multinomial case: It fits a multinomial log-linear model (minimizes the cross-entropy error function).

Also, if needed a **regularization term** could be added to the GLM in order to prevent overfitting. It is used the L_2 regularization, so $\lambda \cdot \|\alpha\|_2$ is added to the error function (MSE or cross-entropy). The value of the λ is trained by CV on the training set (or GCV for the regression case).

Once these αs are found, the training on the SNN is done and the network is ready to be used for predicting new observations. Summarizing, we have seen what is an SNN, how it works, how it is built and how to train it.

4.2 Ensemble of SNNs

The SNNs explained previously are good learners that can deal with mixed data and missing data, but they have some weak points that could be improved. Some of them are:

- An SNN model is very dependent on the number of prototypes and the prototypes selected. For example, selecting a good or bad set of prototypes could create two completely different models, one that works very well and another one that works poorly. Therefore the goodness of fit of the SNNs may have high variance.
- For large problems, the SNN starts to take a lot of time to train the model. Especially if the PAM method is used, where it would have to compute the similarities between all observations in the dataset and do the clustering. Thus the computational cost grows exponentially with the dimension of the problem.

With the main purpose of improving these weak points, it is proposed the use of an ensemble method. In our case, I decided that a Bagging-like method could improve the SNN in the previous points because the fact that it has a set of SNNs will reduce the variance, and each SNN will be trained with a subset of the full dataset, and thus the complexity cost will be reduced for large problems.

4.2.1 General framework

Let us explain how it works and the general framework of the second proposal: Ensemble of SNNs.

As the main idea, the ensemble method uses a set of SNNs instead of just one and combine the predictions of them to give the output of the model. In other words, for each input that the model receives, it gets the prediction of each SNN in the ensemble, and with all these predictions, it is used an ensemble learner to generate the response of the model. Figure 4.3 shows a graphical representation of how it works.

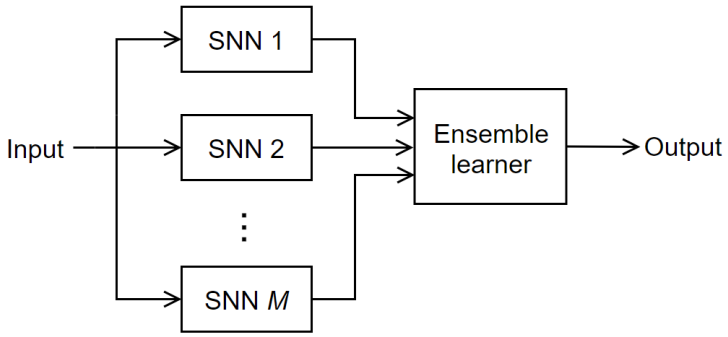

Figure 4.3: Ensemble of SNNs.

The remaining piece of the ensemble to be explained is the ensemble learner. I propose different methods that gather the predictions of the SNNs and compute the output of the model:

- **Method A. Majority vote** (classification) **or mean** (regression).

  This method is the one used by the Random Forest algorithm and consists on, for classification problems, predict the class that was most voted by the SNNs, and for regression problems, it predicts the mean of all SNNs responses. So, the response of the ensemble is:

  Regression:
  $$\hat{y}(x) = \frac{\sum_{i=1}^{M} \nu_i(x)}{M}$$

  Classification:
  $$\hat{y}(x) = \text{majority vote among the } \nu_i(x)$$

  where $\nu_i(x)$ is the output of the ith SNN in the ensemble and M is the number of SNNs in the ensemble.

- **Method A2. Mean of probabilities**.

  This method works only for classification problems and it is very similar to the previous method, but instead of using the majority voting, it does the mean of the probabilities.

- **Method B. Generalized Linear Model**.

  The third method consists of using a Generalized Linear Model (GLM) as the ensemble learner. So, the GLM will have as inputs the responses of the SNNs and as outputs the problem response. In the case of regression, these inputs are the prediction of each network, and in the case of classification, the probabilities are used (and thus for multinomial classification, it will have as many outputs as classes to predict).

  The coefficients of the model (called β_i) are found at the learning stage, and these β_i will be fixed values for all the network (they do not depend on the inputs, they will always be the same coefficients).

  The type of problem to be solved will determine the link function to be used: for regression problems, it is used the identity link function, for binomial cases the *logistic* function, and for multinomial, the *softmax*. So, this ensemble method will predict the response as follows (being β_i the weights of the linear model and ν_j the output of the jth SNN in the ensemble):

  – Regression:

  $$\hat{y}(x) = \psi(x)$$

  – Binomial:

  $$\hat{y}(x) = \frac{1}{1 + e^{-\psi(x)}}$$

  – Multinomial:

  $$\hat{y}_k(x) = \frac{e^{\psi(x,k)}}{\sum_{k2=1}^{K} e^{\psi(x,k2)}}$$

  where (regression and binomial):

  $$\psi(x) = \sum_{j=1}^{M} \beta_j \cdot \nu_j(x) + \beta_0$$

  where (multinomial):

  $$\psi(x,k) = \sum_{k=1}^{K} \sum_{j=1}^{M} \beta_{jk} \cdot \nu_{jk}(x) + \beta_0$$

  The way to train the model and its weights are explained at the learning stage section (Section 4.2.2). As a summary, Figure 4.4 shows a graphical representation of the Ensemble of SNNs with the method B (only the regression case is shown, for the classification cases the *logit* or *softmax* function are missing at the end of the network).

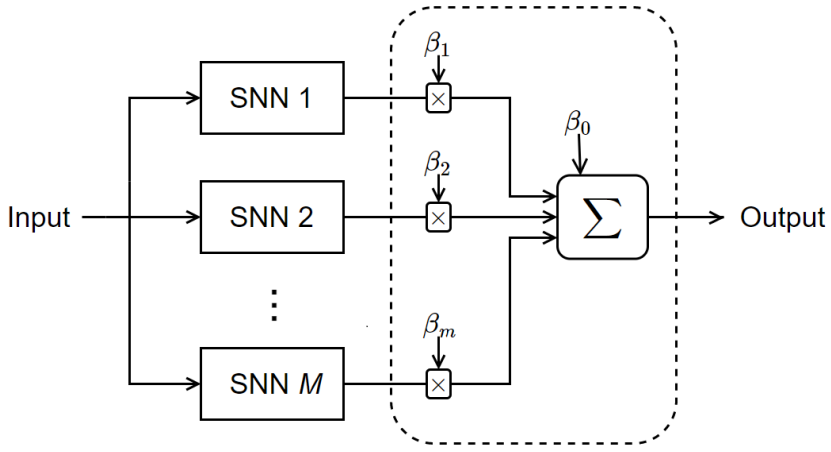

Figure 4.4: Ensemble of SNNs with method B (regression case).

- **Method C. Mixture of Experts (MoE)**

  The Mixture of Experts (MoE) [15] is a technique where the weights (β) of each SNN are determined by the input of the network. In other words, there exists a gating network that given the ensemble input, it decides the weights that will have each SNN. Figure 4.5 shows a graphical representation of this method. Unlike the previous method (where all β were fixed and were computed at the learning stage), the weights are a function of the input ($\beta_i(x)$). So, the ensemble method will predict the response as follows:

$$\hat{y}(x) = \sum_{j=1}^{M} \beta_j(x) \cdot \nu_j(x)$$

  The coefficients $\beta_i(x)$ are determined by the outputs γ_i of the gating network through a *softmax* activation function:

$$\beta_i(x) = \frac{\exp(\gamma_i(x))}{\sum_{j=1}^{M} \exp(\gamma_j(x))}$$

  Thus, the gating network has one output for each SNN and, as a result of applying the *softmax* function, the conditions $\sum_{i=1}^{M} \beta_i(x) = 1$ and $0 \leq \beta_i(x) \leq 1$ are guaranteed. These coefficients $\beta_i(x)$ take the meaning of being the weights for a weighted mean of the outputs of the SNNs. So, the gating network decides the importance of each SNN by looking at the inputs.

  About the gating network, I decided to use a linear model in order to compute each one of the γ coefficients. So, the γ_i function is:

$$\gamma_i(x) = \sum_{j=0}^{P} \Phi_{ij} x_j$$

  where Φ is a matrix of dimensions number of SNNs times the length of x, and thus, $\Phi_{i\cdot}$ are the coefficients of the linear model used to compute the γ_i coefficient.

  As the input of the ensemble could have missing values and mixed data, and the linear model cannot deal with these issues, I decided that the gating network will receive as input the Gower's similarity between x and each one of the prototypes of each SNN. For example, if we have an ensemble with two SNNs, the first one with 3 prototypes (observation 1, 3 and 4)

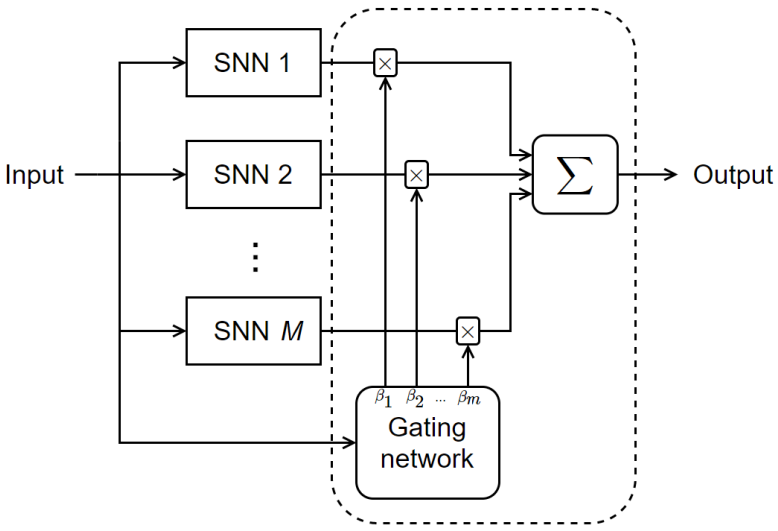

Figure 4.5: Ensemble of SNNs with method C.

and the second with 2 prototypes (observation 5 and 7), then the input of the gating network will have length $3 + 2 = 5$ and the content will be the similarity between x and observations 1, 3, 4, 5 and 7. By making this change and using similarities instead, the gating network can deal with datasets with missing values, and also, the gating network will know the similarities between the input space and the prototypes of each SNN, so this fact could help the network to compute better β coefficients.

In the learning section, how to set these Φ coefficients is explained.

- **Method C2. Mixture of Experts by clustering**

  Finally, the last method uses the same ensemble learner than the Mixture of Experts, but the main difference is in the way each SNN is trained. The main idea is to create an ensemble, where each SNN solves a region of the input space and then the gating network decides for each input the weights to give to each SNN (if the input belongs to the region of an SNN, the network should give higher weight to this SNN than to the others).

  First of all, it is performed a clustering (PAM) on a subset of the observations of at most 5000 observations. It is set a limit of observations because if not, for large datasets this clustering will take a lot of time (the computational cost is exponential). For this reason, we set a limit on the observations to be used for clustering. These observations are sampled randomly from the full dataset. With these observations, it is performed a clustering that finds as many clusters as SNNs in our ensemble (each cluster will be used in one SNN). Then, the clusters are computed for the full dataset (an observation belongs to the nearest cluster medoid).

  Finally, each cluster is used to train an SNN of the ensemble. All other procedure is the same than method C (Mixture of Experts), thus, the main contribution of the method is to try to divide the input space into regions and train an SNN in just one region.

At this point, it is explained how the ensemble of SNNs works, how it is built and the several variants that it has.

4.2.2 Learning stage

Once the general framework is clearly explained, we have to train our model. We are going to explain the steps from left to right (Figure 4.3), so first, it is explained how to train each SNN and then how to train the ensemble learner.

4.2.2.1 Train each SNN

One of the key facts of the ensembles to be a good model is that the set of learners (SNN in our case) should be different between them. For this reason, each SNN should be trained with a different set of observations and also with a different number of observations. As said before, by doing that, we can reduce the complexity of training a single SNN.

The number of observations to train an SNN will be obtained from one of the following methods:

- **Poisson** (P). This method uses the Poisson distribution to get the number of observations.
- **Uniform** (U). This method uses the Uniform distribution to get the number of observations.
- **Constant** (C). All SNN will be trained with the same number of observation.
- **Binomial** (B). This method uses the Binomial distribution to get the number of observations.

The user should set the hyper-parameters of each distribution. And then, the observations to be chosen will be sampled without replacement from the full dataset [2]. Finally, each SNN will be trained with its own subset of observations and the training will be done as explained in section 4.1.2.

Notice that another fact that could add variability among SNNs is when the number of prototypes and the selected prototypes are generated randomly, then each SNN will have different prototypes as neurons.

[2]Except the for ensemble method C2, that each SNN samples the observations from the cluster assigned to this SNN.

4.2.2.2 Train the ensemble learner

Once the SNNs are trained, we have to fit the ensemble learner. Let us see how the different variants are trained:

- **Method A and A2**

  These first methods do not need any training part (there are no parameters to fit), so for these cases, the training consists only on fitting each SNN.

- **Method B**

  This second method is a Generalized Linear Model (GLM), so all β_i are fitted with the training set. For this case, all training data is used, so, first of all, it is needed to compute the response of each SNN for the full dataset and then fit the GLM. This method has a higher computation cost than the previous ones.

  Also, it is interesting to add a **regularization** term when fitting the βs. In this case, the L_2 regularization is used to prevent overfitting (when using regularization, it is called B2 method). A comparison of the effect of this regularization is performed in the experiments section.

- **Method C**

  The Mixture of Experts (MoE) model is trained using optimization techniques. But again, all the training data is used to fit Φ, so it is computed the response of each SNN for the train set.

  Using optimization techniques it is fitted Φ (matrix of dimensions: number of SNNs times the length of x) by minimizing the error function. This error function depends on the type of problem to be solved, these are the three cases:

  - Regression: It minimizes the Sum of Squares Error (SSE).

    $$E(\Phi) = \sum_{n=1}^{N} (y_n - \hat{y}(\Phi, x_n))^2$$

    where y_n is the real response value of observation n and $\hat{y}(\Phi, x_n)$ is the prediction of the ensemble for this observation.

  - Binomial classification: It minimizes the cross-entropy error function for two classes.

    $$E(\Phi) = -\sum_{n=1}^{N} \Big(y_n ln(\hat{y}(\Phi, x_n)) + (1 - y_n) \cdot ln(1 - \hat{y}(\Phi, x_n)) \Big)$$

    where y_n is 1 when the nth observation is positive and 0 if negative.

  - Multinomial classification: It minimizes the cross-entropy error function for K classes.

    $$E(\Phi) = -\sum_{n=1}^{N} \left(\sum_{k=1}^{K} y_{nk} \cdot ln(\hat{y}_k(\Phi, x_n)) \right)$$

    where y_{nk} is 1 when the nth observation is of the k modality, 0 otherwise.

  The optimization technique will find a local minimum of the error function using an iterative procedure. Again, the **gradient descent** method (GD) is used. To solve the optimization, we must provide the objective function (in our case the error function) and the first-order derivatives of it. Appendix A.2 contains all information related to these error functions and all its derivatives for the MoE method.

  The learning of this method ends when the optimization procedure finds a local minimum of the error, and thus we set the Φ matrix with the values found in the optimization. At this point, all parameters of the model are set and the training is completed.

- **Method C2**

  The way to train this method is the same than method C (Mixture of Experts). As said before, the only difference is in the way to sample the observations to train each SNN (clustering).

Summarizing, it was explained the general framework of the Ensemble of SNNs, how the model is built, its main variants and how to train it. Now, we should test the performance of the ensemble by developing experiments and comparing the different methods designed in this section.

Chapter 5

Implementation and Use

In this section, it is briefly explained how I have implemented the SNN and the Ensemble of SNNs. First of all, I decided to use the *R* language because it is widely used for developing statistical software and data analysis, and it fits on the thesis purpose.

I made an implementation that tries to encapsulate all functionalities of the learners inside a function, and a data scientist only has to use these functions without knowing its implementation. So, the objective of it was to create like an *R* package that has all SNN functionalities.

Let us see how to use these functions and how to create and fit the learners. There are two main functions: one that creates the SNN model and another for creating the Ensemble of SNNs. The following function fits an SNN model given a dataset:

```
snn(Target ~ . , dataset, ... )
```

And this second function creates an Ensemble of SNNs:

```
snn.bagging(Target ~ . , dataset, ... )
```

These functions return the fitted object corresponding to the model. The three dots in these functions are optional parameters that can be added to the call in order to use one variant of the model and configure it as desired (e.g. setting the ensemble learner method).

And finally, the following function uses the fitted model and predict the response for new observations:

```
predict(fittedObject, newdata)
```

As said before, it is tried to do a user-friendly implementation and to encapsulate all the learners in one function to create the model and another one to predict new data.

Also, I want to stress the fact that I have **optimized** the code as much as I could because if not, for large problems, it could take a lot of time to be trained (days) or it could simply fail due to memory limitation issues.

One of these optimizations is that I have dug in some *R* packages and I have modified its source code in order to make some changes on them and improve its performance. Specifically, I modified the source code of the *cluster* package and I changed its *R* and *fortran* files. In this package, there is a function, called *daisy*, that computes the Gower's similarity for a set of observations. This function was designed for clustering purposes and it computes the similarity between all pairs of observations in the dataset (if the purpose is to find clusters in the data, this function is okay). But for the SNNs, what we want is the similarity between a set of prototypes and the input data of the network (e.g. train set or test set). So, the *daisy* function will compute a lot of unnecessary similarities that consumes a lot of CPU time (e.g. similarities between prototypes or between observations in the input set). Moreover, for large problems this issue is even worse. For example, for a dataset with 10^5 observations *daisy* computes $10^5 \cdot (10^5 - 1)/2$ similarities, and most of them are not used in the SNN. In my local computer, for large problems, the *daisy* gets an error because it cannot allocate that amount of memory. The only solution for this problem was to modify the source code of the *daisy* function. So, the code has been modified to allow the computation of similarities between two different sets of observations.

Moreover, as seen in Section 3.1.2, the similarity measures set some of their parameters from the sample data. For example, the similarities of numerical variables divide by the range of the variables, and this range is computed from the sample set. So, I have also modified the *daisy* function to return all these similarity parameters (e.g. range, modalities, ...) and then, these values are used to compute similarities for new data.

With these modifications, I have adapted the *daisy* function for SNN purposes, and without them, large problems could not have tried.

In the following section, this implementation is used to run the experiments and see the behaviour of the SNNs.

Chapter 6

Experiments

Until now, it is explained, designed and implemented the learning algorithms proposed in this thesis, and now we should test how they perform. So, in this section, the experiments that were designed to test the behaviour of the SNN and the Ensemble of SNNs are explained.

6.1 Datasets

First of all, I introduce the chosen problems (or datasets) that are used to do the experiments. For our case, as the SNN can deal with regression, binomial and multinomial classification problems, it is selected a few of each type. Also, we have chosen datasets of different sizes (at least a large dataset for each type of problem, with more than 40 000 observations), datasets with different types of variables (numerical, categorical, binary,..) and some with missing values. Most of these datasets are classical problems and were found in the UCI Machine Learning Repository.
The chosen datasets are:

Regression problems:

- **Automobile Dataset** . This first dataset contains information related to cars and the goal is to predict the price that was paid for it. It has 205 observations, 25 attributes (14 numeric, 8 nominal and 3 ordinal) and it has missing values.

  `https://archive.ics.uci.edu/ml/datasets/Automobile`

- **Auto MPG Dataset**. This dataset contains information about automobiles and the main goal is to predict the *mpg* attribute (fuel consumption in miles per gallon). It has 398 observations, 7 numeric attributes and it contains a few missing values.

  `https://archive.ics.uci.edu/ml/datasets/Auto+MPG`

- **Communities and Crime Dataset**. This dataset combines socio-economic data from the 1990 US Census, law enforcement data from the 1990 US LEMAS survey, and crime data from the 1995 FBI UCR. The goal is to predict the total number of violent crimes per 100K population. It has 1994 observations, 122 attributes (121 numerical and one ordinal), and it has more than 15% of missing values.

  `https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime`

- **MV Dataset**. This dataset contains artificial data that was generated with dependencies between the attribute values (for more information go to the dataset source). This dataset has 40 768 observations and 10 attributes (7 numerical, 2 categorical and 1 binary variable).

  `https://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html`

Binomial classification problems:

- **Statlog (Heart) Dataset**. The first classification dataset contains information related to a set of patients and the goal is to predict the presence or absence of heart disease. It has 270 observations and 13 attributes (6 numerical, 3 nominal, 1 ordinal and 3 binary variables).

  `https://archive.ics.uci.edu/ml/datasets/Statlog+%28Heart%29`

- **Horse Colic Dataset V1 (binomial case)**. This dataset contains information about horses with a colic disease. There are several possible goals for this dataset, but for the binomial classification, the objective is to predict if the lesion was surgical. It has 368 observations, 21 attributes (7 numerical, 3 categorical, 8 ordinal and 3 binary) and more than 25% of missing values.

  `https://archive.ics.uci.edu/ml/datasets/Horse+Colic`

- **Pima Dataset**. The goal of the Pima dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements. It has 768 observations, 8 numerical attributes and missing values.

  `https://www.kaggle.com/uciml/pima-indians-diabetes-database`

- **Mammographic Mass Dataset**. It is a dataset for the classification of benign and malignant mammographic masses based on BI-RADS attributes and the patient's age. It has 961 observations, 4 attributes (1 numeric, 2 nominal and 1 ordinal) and it has missing values.

  `https://archive.ics.uci.edu/ml/datasets/Mammographic+Mass`

- **Mushroom Dataset**. This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. The goal is to classify into an edible or poisonous mushroom. It has 8124 observations, 21 attributes (1 numeric, 17 nominal and 3 logical) and it has missing values.

  `https://archive.ics.uci.edu/ml/datasets/Mushroom`

- **Census-Income (KDD) Dataset**. This dataset contains weighted census data extracted from the 1994 and 1995 current population surveys conducted by the U.S. Census Bureau and the goal is to predict if the income of a person is higher or lower than 50 000$. The dataset has 299 285 observations, 41 demographic and employment-related variables (8 numerical, 28 categorical and 5 binary) and it has 24% of missing values.

  `https://archive.ics.uci.edu/ml/datasets/Census-Income+%28KDD%29`

Multinomial classification problems:

- **Audiology Dataset**. This dataset contains data related to hearing diseases. The original dataset has more than 20 classes, but they have been grouped into 4 main groups. It has 226 observations, 31 attributes (1 categorical, 7 ordered and 23 logical variables) and it has missing values.

  `https://archive.ics.uci.edu/ml/datasets/Audiology+%28Standardized%29`

- **Glass Dataset**. The goal of this dataset consists of making a classification into one of the six types of glasses based on chemical information about them. The dataset has 214 observations and 9 numerical attributes.

  `https://archive.ics.uci.edu/ml/datasets/Glass+Identification`

- **Horse Colic Dataset V2 (multinomial case)**. This dataset is the same than the binomial classification case, but in this case, the goal is to predict what eventually happened to the horse (lived, died or was euthanized).

  `https://archive.ics.uci.edu/ml/datasets/Horse+Colic`

- **Annealing Dataset**. This dataset contains information related to steel annealing and the goal is to classify each observation into one of the five classes. It has 798 observations, 31 attributes (9 numerical, 5 categorical, 1 ordinal and 16 binary) and more than 60% of the data is missing.

  `https://archive.ics.uci.edu/ml/datasets/Annealing`

- **Contraceptive Dataset**. The objective of this dataset is to predict the current contraceptive method choice (no use, short term or long term methods) of a woman based on her demographic and socio-economic characteristics. It has 1473 observations and 9 attributes (2 numerical, 1 categorical, 3 ordinals and 3 binary).

  `https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice`

- **Diabetes 130-US hospitals for years 1999-2008 Dataset**. This dataset contains information about patients with diabetes and the goal is to predict if a patient will be readmitted more than 30 days in the hospital, less than 30 days or will not be readmitted. It has 101 766 observations, 45 attributes (8 numeric, 9 categorical, 25 ordered and 3 binary) and it has 8% of missing values.

  `https://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008`

Table 6.1 shows a summary of the datasets selected for the experiments of this thesis.

Dataset	**#Obs.**	**#Attr.**	**Data types**	**Missing**	**#Classes**
Regression					
Automobile	205	25	14R, 8N, 3O	1%	-
AutoMPG	398	7	7R	0.2%	-
Communities	1994	122	121R, 1O	15.1%	-
MV	40 768	10	7R, 2N, 1B	0%	-
Binomial classification					
Heart	270	13	6R, 3N, 1O, 2B, 1A	0%	2
HorseColicV1	368	21	7R, 3N, 8O, 3B	27.1%	2
Pima	768	8	8R	10.6%	2
Mammographic	961	4	1R, 2N, 1O	4.1%	2
Mushroom	8124	21	1R, 17N, 2B, 1A	1.5%	2
Census	299 285	41	8R, 28N, 5B	24.1%	2
Multinomial classification					
Audiology	226	31	1N, 7O, 23A	2.1%	4
Glass	214	9	9R	0%	6
HorseColicV2	368	21	7R, 3N, 8O, 3B	27.1%	3
Annealing	798	31	9R, 5N, 1O, 4B, 12A	60.2%	5
Contraceptive	1473	9	2R, 1N, 3O, 3B	0%	3
Diabetes	101 766	45	8R, 9N, 25O, 1B, 2A	8.1%	3

where data types are codified as: (R)eal, (N)ominal, (O)rdinal, (B)inary and (A)symmetric binary

Table 6.1: Characteristics of the datasets used in the experiments.

6.2 Methodology

At this point, I have chosen the set of problems that will be used to perform the experiments. But before explaining them, a generic methodology is defined that all experiments should follow and it should be clearly explained.

First of all, in order to test a model, the **holdout method** is used: the dataset is divided into two sets (train and test), the train set is used to create and train the model and the test set will be used to test this model with data that was not seen in the training stage. I decided to use the classical split ratio, so 2/3 of observations are used to select and create the best model and the remaining 1/3 to test it.

Also, just as a remember (it was explained in Section 4), in some cases, a **10-fold cross-validation on the train set** is performed to set the parameters of the model (e.g. regularization term or p for the CV method).

The train and test split of the data could be very determinant when testing a model, two different splits can give very different performance metrics. For this reason, this procedure is repeated **50 times**, so it is done 50 splits into train and test and each split is used to train the model and test it. By doing this repetition we have a better view of how the method performs on the current problem, and graphically, we can show boxplots with the metrics. For large problems, it is reduced the number of repetitions because the execution time of running these experiments was very high.

Most of the experiments on this thesis consist of comparing different methods (e.g. SNN vs decision tree) or different variants of a method in order to see its behaviour. For these type of experiments, the *holdout* method is executed 50 times, as explained before, but each method does the same splits of the data (i.e. the ith split is the same for all methods in the experiment). By doing this way, all methods are tested with the same data.

Also, as said before, the selected datasets are very different among them and many have missing data and mixed data. This fact makes that most of the classical learners fail when trying to solve the problem. Two of the learners that can deal with these issues are: decision tree and Random Forest. By similarity on the way to build the models, I decided to compare the Ensemble of SNNs with the Random Forest and the SNN with a decision tree. The Random Forest [16] is a learning algorithm consisting of an ensemble of decisions trees, and it uses bagging and feature selection to create different and uncorrelated trees. The main idea behind this method is that the prediction by committee of the decision trees is more accurate than any of these trees alone. So, these methods will be used as benchmarks for the experiments.

And the last thing to mention is that the experiments explained below try and test several configuration/variants of the SNN and the Ensemble of SNNs in order to analyze the influence of them. But all experiments do not try all possible configurations, usually, they test a set of variants (the ones that are the case of study) and all other parameters are fixed for all the experiment. For this reason, there is the need to define a default model for the SNN and the Ensemble. Then the experiment will use these default values for the parameters that are not the case of study. Let us define these default values:

- **SNN**

  By default, the number of prototypes of the network is obtained from a Poisson distribution that has an expected value equal to the 10% of the length of the training set. The prototypes are chosen by the PAM clustering method. The value of p of the activation function is set by the optimization procedure. And the GLM does not use regularization when training the model.

- **Ensemble of SNNs**

  By default, it is used the A ensemble learner and an ensemble with 100 SNNs. The number of observations used to train each SNN is sampled from a Poisson distribution that has an expected value of the 50% of the training set. The SNNs are trained with the default values explained above except for the prototypes selection that they are selected randomly instead of PAM. And also, for large problems, the expected values of the Poisson distributions are reduced in order to reduce the computational cost.

As said before, this is just the default configuration of the models and each experiment will play with the parameters that are analyzed.

6.3 Experiments

To test the behaviour of the proposals of this thesis, it is designed a set of experiments. The first two experiments are the most important and exhaustive ones, I call them *main experiments*. With them, I analyze how the SNN and the Ensemble of SNNs perform on all studied datasets (small and large), and a comparison is made among the main variants of each model. With these experiments, I get the main conclusions of this thesis and also they are the ones that take longer to be executed. Secondly, there are the *complementary experiments*. This second group are the experiments that test a specific feature and they are run only for small problems (large are excluded because of execution time cost).
Let us explain the experiments:

Main experiments:

As said before, these are the most important experiments of this thesis because they analyze the behaviour of the SNN and the Ensemble of SNNs for small and large datasets, and it is tested most of the configurations of each method.

- **Experiment 1: Comparison of several variants of the SNN and the decision tree**.

  The goal of the first experiment is to compare the results obtained with different variants of the SNN and the results obtained with the decision tree, that will be used as the benchmark learner. This experiment is focused on analyzing the SNN learning method.

  The main variants of the SNN that are analyzed in this experiment are:

  - The clustering method used to select the number of prototypes of the SNN. This method can be:
    - PAM
    - Random
  - The method used to set the p value of the activation function (f_p). There are four methods:
    - Constant value ($p = 0.1$)
    - Optimization procedure
    - Cross-validation
    - Generalized cross-validation (only for regression problems)
  - Whether the regularization is used on the linear model.

  For each problem, I run all possible combination of the previous variants (some are not possible, e.g. setting the p with the optimization procedure and using regularization). Also, because of execution time issues, the PAM and the cross-validation methods are excluded for problems with more than 5000 observations (these methods are very expensive and for large problems, it could take days to create the model). For the other parameters of the SNN that are not mentioned above, it is used the default values of them.

So, with this experiment we are going to get an overview of the behaviour of the different configuration of the SNN for different datasets, we will see the methods that perform better and we will compare these results with a popular learner (decision tree).

- **Experiment 2: Comparison of several variants of the Ensemble of SNNs and Random Forest**.

  For the second experiment, a comparison among the main variants of the Ensemble of SNNs is made, taking as a benchmark the results of the Random Forest. This experiment is focused on analyzing how the Ensemble of SNNs performs.

The main variants that are analyzed in this experiment are:

- The ensemble learner: A, A2 (only for classification), B, C and C2.
- Whether the regularization is used on the linear model (ensemble learner).

Again, all problems are tested for this experiment, and it is run all possible combination of the previous methods. But only the ensemble method B allows regularization, let us call B2 to the B method with regularization (and simply B the one without regularization). The default configuration is used for the way to build the SNNs of the ensemble and the other parameters that were not mentioned above.

The results of this experiment will give us an overview of the behaviour of the different variants of the Ensemble of SNNs, the methods that perform better and we will compare these results with the Random Forest.

Complementary experiments:

The *complementary experiments* are experiments that only test a specific feature or configuration of a learner (SNN or ensemble), an example could be the number of prototypes of the network. For these experiments, it is set the default configuration for the SNNs and the ensembles, and it is played/modified the analysed feature in order to see its influence on the model. Also, as they are complementary to the *main experiments* and because of execution time issues, the large problems are not tested and, for each experiment, 20 repetitions are executed.

- **Experiment 3 (SNN): Analysis of the number of prototypes of a single SNN**.

This experiment analyzes the influence that the number of prototypes has on the performance of a single SNN. To do that, it is tested several percentages (from 0% to 100%) of observations that will be used as prototypes. With this experiment, I want to understand the behaviour of the network for several numbers of prototypes and answer questions like the following ones: which is the most suitable percentage of data to be used as prototypes? Which is the minimum percentage that does not loss goodness of fit? Do high percentages fail by overfitting?

- **Experiment 4 (Ensemble): Comparison of PAM and Random prototype selection for the Ensemble of SNNs**.

The fourth experiment analyzes if the clustering methods used to select the prototypes of the SNNs has some influence on the performance of the Ensemble. The two clustering methods that are analyzed are PAM and Random.

- **Experiment 5 (Ensemble): Analysis of adding regularization to each SNN of the Ensemble.**

With this experiment, it is tested if adding regularization to each one of the SNNs of an Ensemble of SNNs performs better or worse. So, for each problem, it is done a comparison between the performance of the Ensemble when regularizing the SNNs and when they are not regularized.

- **Experiment 6 (Ensemble): Analysis of the distribution used to select the number of prototypes of each SNN of an Ensemble of SNNs.**

This experiment consists of testing the influence of the distribution that decides the number of prototypes to be used in each SNN of an Ensemble of SNNs. As explained before, the number of prototypes to choose can be sampled from one of the following distributions: Uniform, Poisson, Binomial and Constant. So, we are going to analyze the distribution that performs better.

- **Experiment 7 (Ensemble): Analysis of the method used to select the number of observations to train each SNN of an Ensemble.**

  Each SNN of an Ensemble of SNNs is trained with a different set of observations, and the length of these sets (number of observations) is generated by one of the following distributions: Uniform, Poisson, Binomial and Constant. This experiment consists on analyzing if the choice of the distribution that decides the number of observations has an impact on the performance of the model and find if some of these distributions do the task better or worse than the others. For this experiment, it is set that all these distributions have the same expected value. For this case, it is set that the expected number of observations is 50% of the training set.

- **Experiment 8 (Ensemble): Analysis of the proportion of data used to train each SNN of an Ensemble.**

  This experiment is very related to the previous one, but instead of analyzing the influence of the distribution, in this experiment, it is studied the number of observations used to train each SNN. To do that, several percentages (between 0% and 100 %) of the training set are tested. So, we analyze, for example, if training each SNN with 10% performs better or worse than using 50%. For this case, the constant distribution is used because we want all SNNs to be trained with the same number of observations.

- **Experiment 9 (Ensemble): Missing values analysis.**

  Finally, the last experiment consists of analyzing the impact of missing values in our dataset. To do that, it is randomly set to *NA* a certain percentage of the total training data. The goal of this experiment is to understand how robust is the Ensemble of SNNs against the missing values.

Chapter 7

Results

In this chapter, the results of the experiments explained in Section 6 are presented. Also, there is a brief discussion of these results. To test the models and compare them, the performance metrics used are the Normalized Root Mean Square Error (NRMSE) for regression problems and accuracy for classification (Section 3.7). Of course, the performance metrics are computed with the test set.

For the *main experiments*, the results obtained for each dataset are briefly commented (showing boxplots) and at the end, there is a global comment of the experiment. For the *complementary experiments*, the analysis is more globally and every dataset is not commented on so many details.

7.1 Experiment 1

Comparison of several variants of the SNN and the decision tree

The first experiment does a comparison among several variants of SNNs and the decision tree. The parameters that are tested are most of the SNN's and they are fully explained in Section 6.3. For each problem, the results are shown through boxplots [1] of the performance metric and the execution time. Appendix B.1 contains all numerical results of this experiment. First, it is shown the regression problems, then the binomial classification and finally the multinomial classification problems.

7.1.1 Regression problems

Automobile dataset

Figure 7.1 shows the results obtained with the Automobile dataset. In this case, all executions of the SNN scores lower NRMSE than the decision trees, so the SNN seems to perform better for this problem. Also, there is not a lot of differences among the different methods of SNN, but it seems that the random prototype selection performs better than the PAM and that the cross-validation methods to choose p slightly improve the other methods. About the execution time, as expected, the cross-validation method needs much more time to create the model than the others.

AutoMPG dataset

Figure 7.2 shows the results for the AutoMPG dataset. Again, the SNN improves the results of the decision trees. The boxplot is very similar to the previous dataset, but in this case, PAM seems to work slightly better than the Random. Also, notice that the worst cases are obtained when the p is constant, and all other methods to choose the p obtain similar results. In this case, the regularization does not have a high impact on the performance metric, but it seems that it gets a bit better results than without regularization. About the execution time, again the cross-validation

[1] In these boxplots, *Rand* means Random, *Const* means constant, *Opt* means optimization procedure, *CV* means cross-validation, *GCV* means Generalized cross-validation, *Reg* means regularization and *Tree* means decision tree.

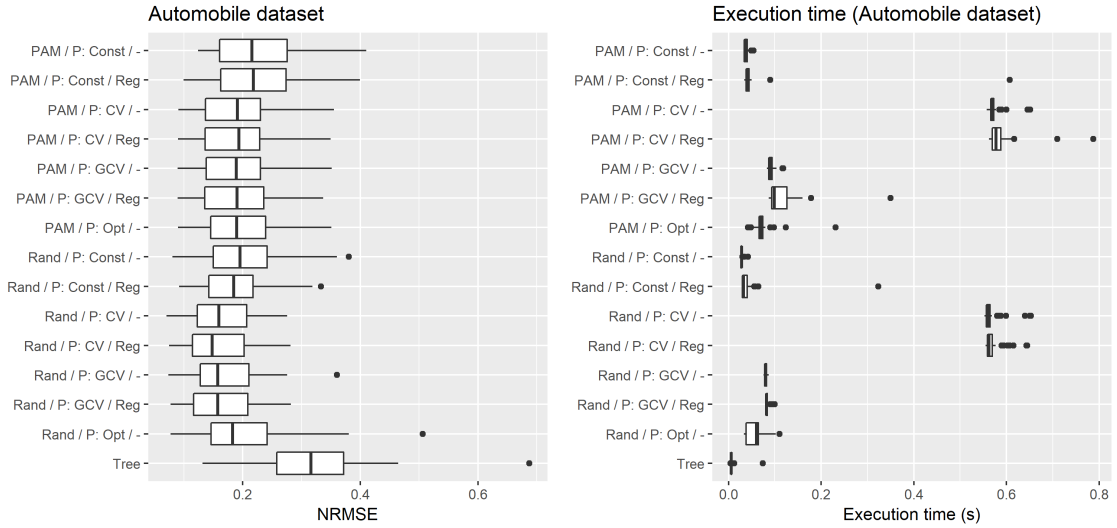

Figure 7.1: Results of experiment 1 for the Automobile dataset.

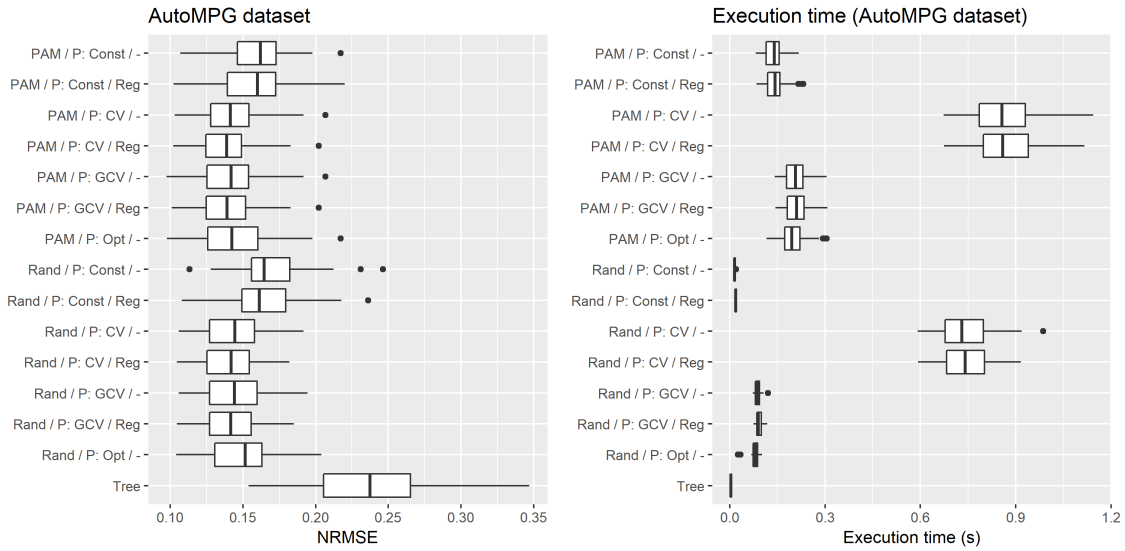

Figure 7.2: Results of experiment 1 for the AutoMPG dataset.

takes much more time than the other methods, and the constant seems to be the faster method (as expected).

Communities dataset

Figure 7.3 shows the results obtained with the Communities dataset. The SNN models improve a lot the NRMSE obtained with the decision tree. This dataset has a different behaviour than the previous ones, the regularization has a high impact in the performance of the model, and the prototype selection and p method do not seem to have a high influence. About the execution time, as this problem has more observations than the previous (more than 1000), the use of PAM starts to penalize the learning time (this clustering method works poorly in terms of execution time for large datasets). In the other hand, the random seems to be much faster and obtain similar NRMSE. About the value of p, again, the cross-validation needs more time than the other methods.

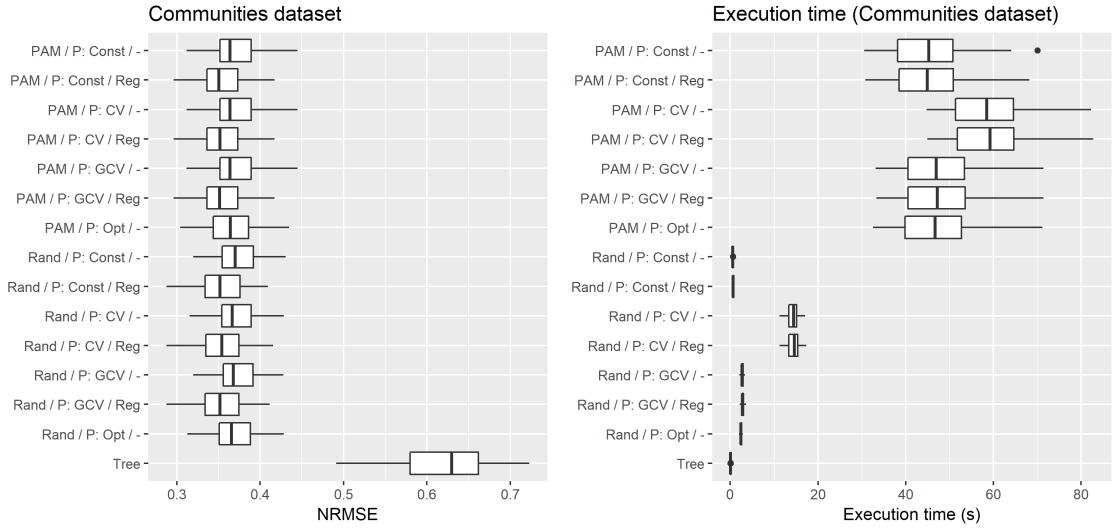

Figure 7.3: Results of experiment 1 for the Communities dataset.

MV dataset

Finally, a large dataset is tested. Because of executing time issues, the number of repetitions is reduced to 20 and the cross-validation and PAM methods are excluded for this dataset. Figure 7.4 shows the results obtained. All models obtain a very low NRMSE, but the SNNs are the ones that obtain better results. The mean NRMSE of the SNNs are very similar among them, but it seems that the optimization procedure has a few executions that are far from the mean. Despite that, all three methods obtain good results. About the execution time, the decision tree is much faster than the SNNs, and the p constant without regularization is faster than the other two methods.

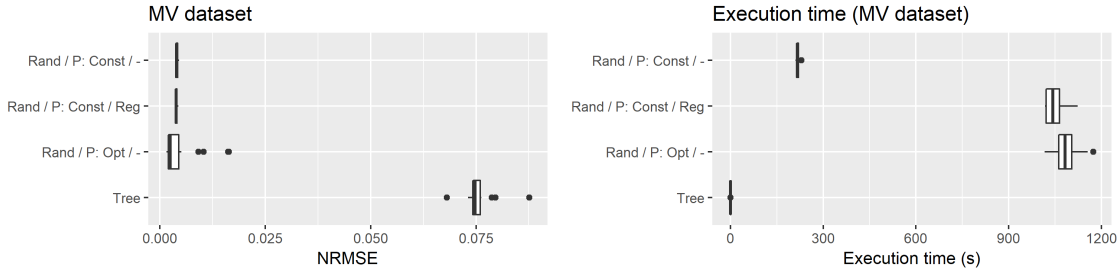

Figure 7.4: Results of experiment 1 for the MV dataset.

7.1.2 Binomial classification

Heart dataset

Figure 7.5 shows the results of the Heart dataset. For the first binary classification problem, the results obtained are quite good. All the variants of the SNNs improve the accuracy obtained with the decision trees. For this case, adding regularization to the network increases the accuracy of the model a little bit. The clustering and p methods do not have a high impact. About the execution time, again the cross-validation cases are slower than the others.

Horse Colic dataset (binomial case)

Figure 7.6 shows the results of the Horse Colic dataset. This dataset has a behaviour very similar to the Heart dataset. Again, the SNNs get better performance metrics than the decision tree. It seems that the factor that increases more the accuracy is the regularization. Also, the execution time of the experiments is very similar to the previous dataset.

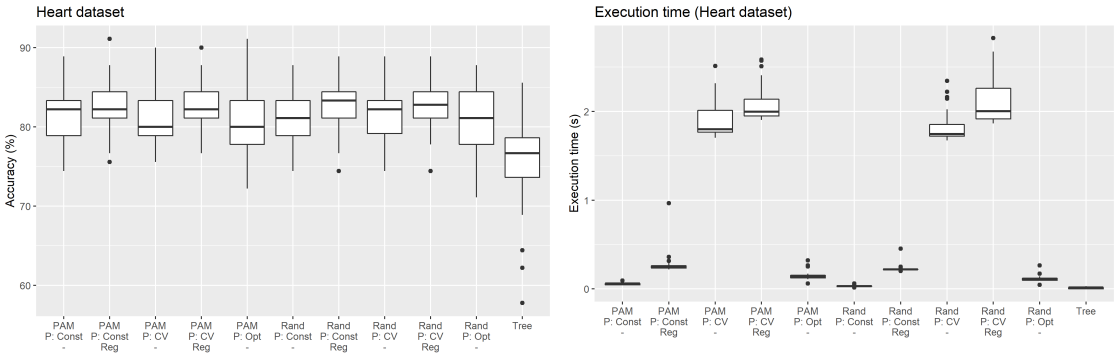

Figure 7.5: Results of experiment 1 for the Heart dataset.

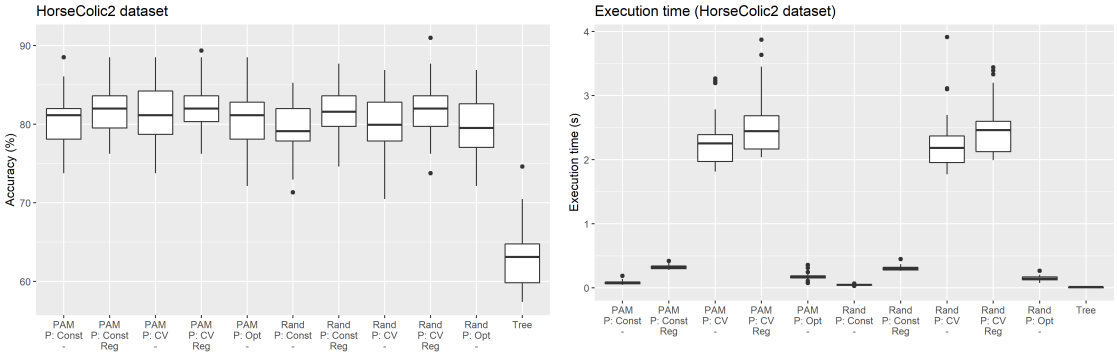

Figure 7.6: Results of experiment 1 for the Horse Colic dataset (binomial case).

Pima dataset

Figure 7.7 shows the results of the Pima dataset. Again, the decision tree obtains lower accuracy than the SNNs and adding regularization to the linear model seems to increase the accuracy of the model.

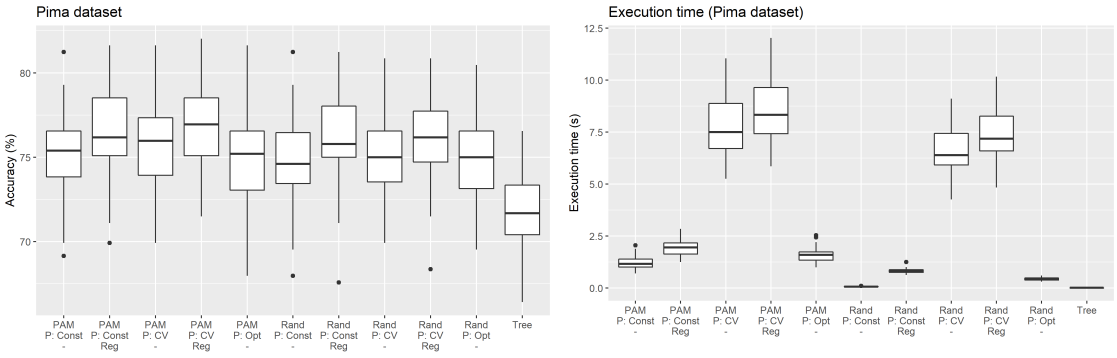

Figure 7.7: Results of experiment 1 for the Pima dataset.

Mammographic dataset

Figure 7.8 shows the results of the Mammographic dataset. Again, we can extract the same conclusions than with the previous datasets, so, it seems that the regularization has a high influence on the model accuracy.

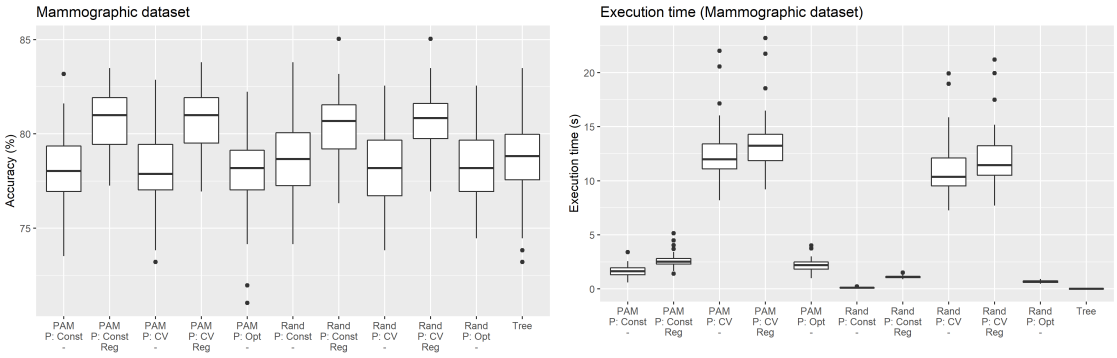

Figure 7.8: Results of experiment 1 for the Mammographic dataset.

Mushroom dataset

Figure 7.9 shows the results of the Mushroom dataset. This dataset has more observations than the previous problems (more than 5000) and the cross-validation and PAM are not executed because of training time issues. In this case, unlike before, the regularization gets lower accuracy than the other methods (p chosen by the optimization procedure and the constant without regularisation get nearly 100% of accuracy). In this case, it seems that the optimization procedure is the method that better performs. About the execution time, the variance is high because depending on the number of prototypes used in the network, it could make the problem to be very slow.

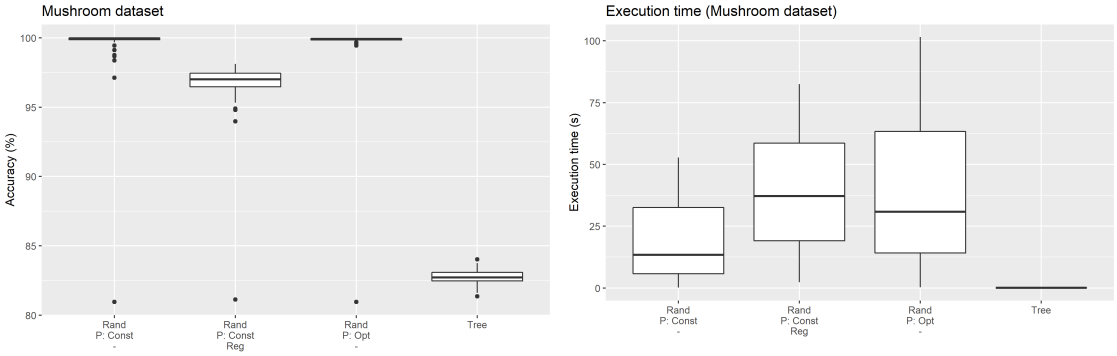

Figure 7.9: Results of experiment 1 for the Mushroom dataset.

Census dataset

Finally, a large binomial classification problem is tested: the Census dataset. For this case, I run only 5 repetitions of each experiment because each one takes a lot of time (some executions take more than 1 hour). Figure 7.10 shows the results of this experiment. With this dataset, it is seen a similar behaviour than the Mushroom dataset. The regularization decreases the accuracy, so the highest accuracies are obtained without regularization and with p constant or optimized. And about the execution time, it is obtained the expected results: the regularization method spends more time to train the model, then the optimization of p, followed by the constant value of p and the decision tree is the fastest method.

7.1.3 Multinomial classification

Audiology dataset

Figure 7.11 shows the results of the Audiology dataset. The SNNs obtain higher accuracies than the decision tree. Also, again, adding regularization increases the accuracy of the model. And for the execution time happens the same than with the previous datasets, the cross-validation is the method that needs more training time.

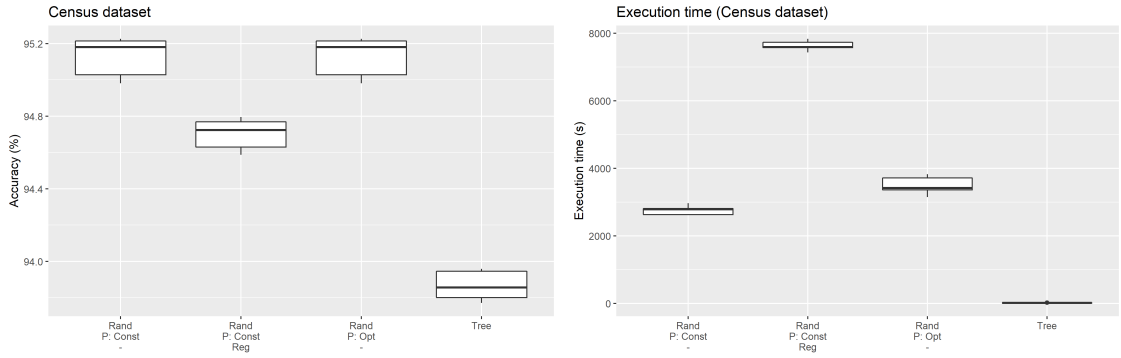

Figure 7.10: Results of experiment 1 for the Census dataset.

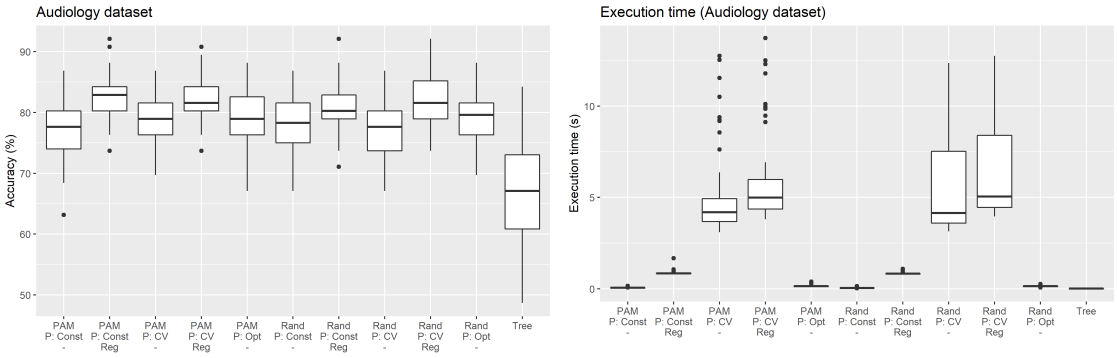

Figure 7.11: Results of experiment 1 for the Audiology dataset.

Glass dataset

Figure 7.12 shows the results of the Glass dataset. This case is the opposite than before, adding regularization decreases the accuracy of the model. Also, mention that in this case, the decision tree performs similar than the SNN without regularization.

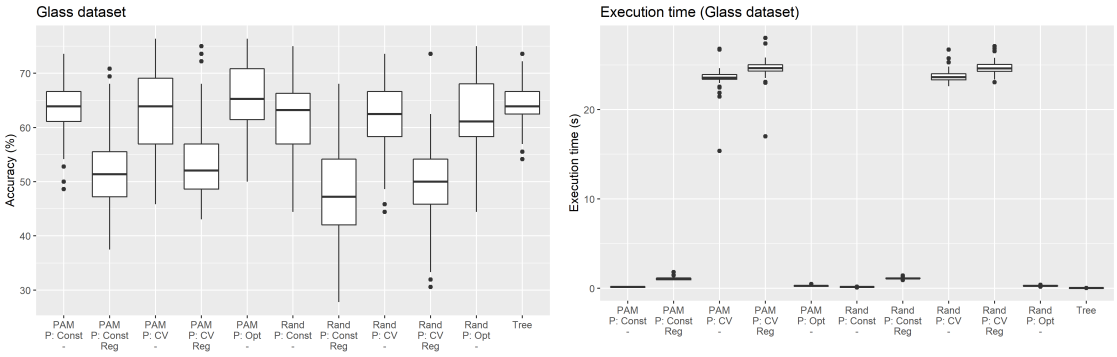

Figure 7.12: Results of experiment 1 for the Glass dataset.

Horse colic dataset (multinomial case)

Figure 7.13 shows the results of the Horse colic dataset. For this problem, the regularization increases the accuracy of the model. These results are very similar to the ones obtained with the Audiology dataset.

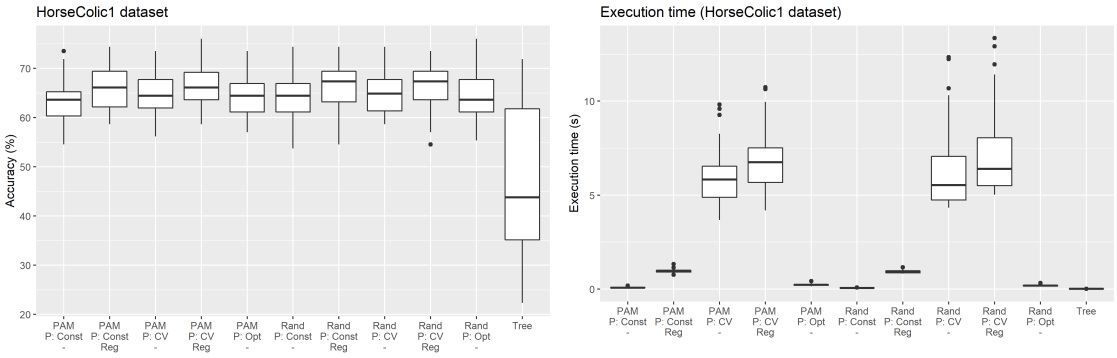

Figure 7.13: Results of experiment 1 for the Horse colic dataset (multinomial case).

Annealing dataset

Figure 7.14 shows the results of the Annealing dataset. The boxplot shows that the optimization procedure performs better than all other methods, included the decision tree. About the execution time, when it is used the optimization of p or it is set the p constant the execution time is quite faster compared to the other SNNs variants (especially the cross-validation ones).

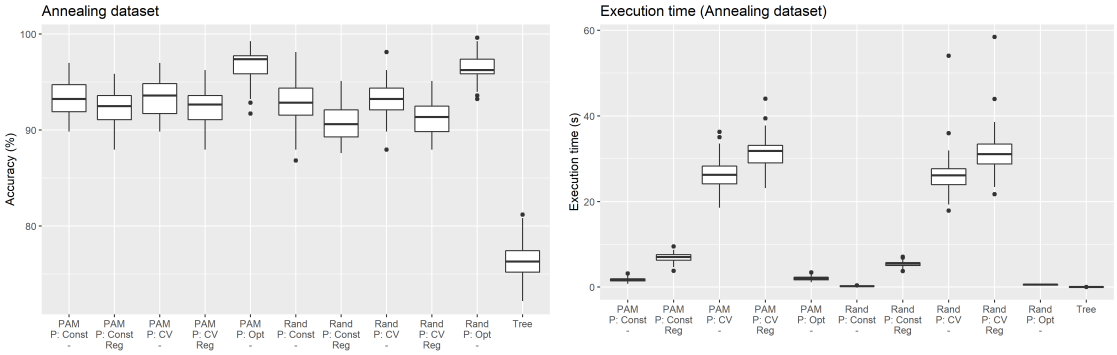

Figure 7.14: Results of experiment 1 for the Annealing dataset.

Contraceptive dataset

Figure 7.15 shows the results of the Contraceptive dataset. Again, adding regularization to the network increases the accuracy of the model. For this problem, the results are very similar to the decision tree, but when using regularization these accuracies are improved a little bit. The execution time boxplots are very similar to the previous datasets.

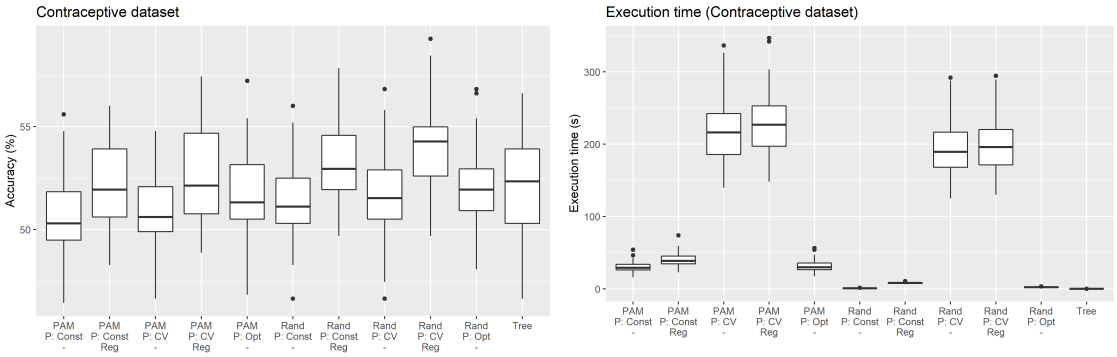

Figure 7.15: Results of experiment 1 for the Contraceptive dataset.

Diabetes dataset

Finally, the network is tested with a large problem. For this case, I run just 5 repetitions of each experiment because each repetition takes much time. Figure 7.16 shows the results of this dataset. All SNN methods get similar results, having the one with regularization a slightly better accuracy than the others. About the execution time, adding regularization to the SNN increases the training time of the network. Moreover, notice that for this large problem the decision tree spends a lot more time to train the model than the SNNs.

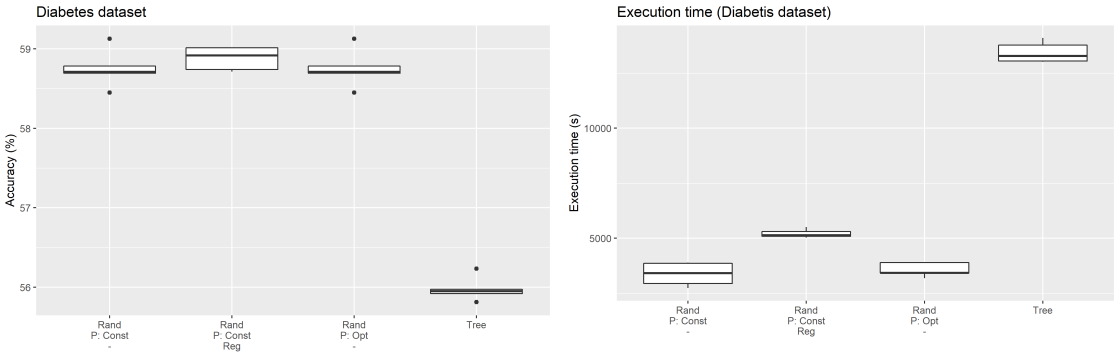

Figure 7.16: Results of experiment 1 for the Diabetes dataset.

7.1.4 Summary of Experiment 1

Summarizing, with Experiment 1 we have seen that our implementation of the SNN improves the accuracy/NRMSE of the decision tree in most of the cases. Among the several variants of the SNN, adding regularization to the linear model seems to be the variant that has the highest influence on the performance metric, it usually improves it notably (but in very few cases, this regularization makes the model worse). About the prototype selection method, there is not a clear winner, in some cases, the PAM gets better results, and in others the random. Finally, the way to choose the p of the activation function does not have a huge impact on the performance metric, but it improves the model a little bit, it seems that the cross-validation method gets the best results, and the constant value the worst. I also want to mention that as the optimization procedure of p does not allow regularization, this method obtains lower results than the ones with regularization.

About the execution time, in almost all problems, the decision tree is much faster than the SNN. About the SNNs, the cross-validation way to choose p needs much more time than the other methods, and the constant is the fastest one. About the clustering method, the Random is faster than the PAM. And adding regularization to the network increases the training time too.

7.2 Experiment 2

Comparison of several variants of the Ensemble of SNNs and Random Forest

The parameters that are tested in this experiment are the ensemble learner and whether it uses regularization (more details in Section 6.3). For notation purposes, I will call B2 to the ensemble learner with the B method and with regularization. As done in the previous experiment, it is shown the boxplots of the performance metric and the execution time. Appendix B.2 contains all numerical results of this experiment.

7.2.1 Regression problems

Automobile dataset

Figure 7.17 shows the results of the Automobile dataset. For the first problem, we can see that the Ensemble of SNNs (or EnsSNN) performs better when it uses the A method. In the other hand,

the B and C2 methods perform very bad for this dataset, but it is interesting to mention that, adding regularization to the B method (B2), these results are highly improved, so, that suggests that the B method has failed by overfitting. The Random Forest is clearly the winner for this dataset, but it is not very far from the A method. Finally, about the execution time, among the EnsSNN methods, the C and C2 need more time to train the model than the others, and the A is the fastest one. Also, the Random Forest is faster than all EnsSNN methods.

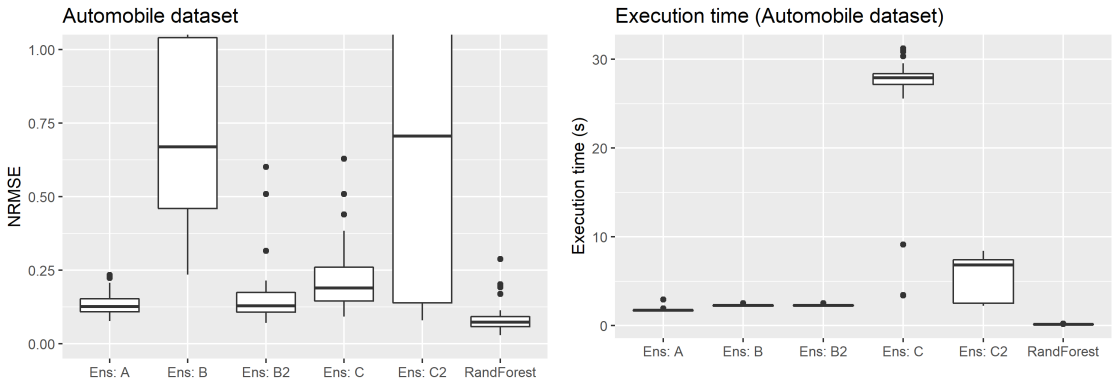

Figure 7.17: Results of experiment 2 for the Automobile dataset.

AutoMPG dataset

The second regression dataset has a similar behaviour than the previous one, the Random Forest is the method that performs better, and among the variants of EnsSNNs, A and B2 methods obtain the best results and the B and C2 are the worst cases. About the execution time, it happens the same than before too, C and C2 spend more time for training than the other methods.

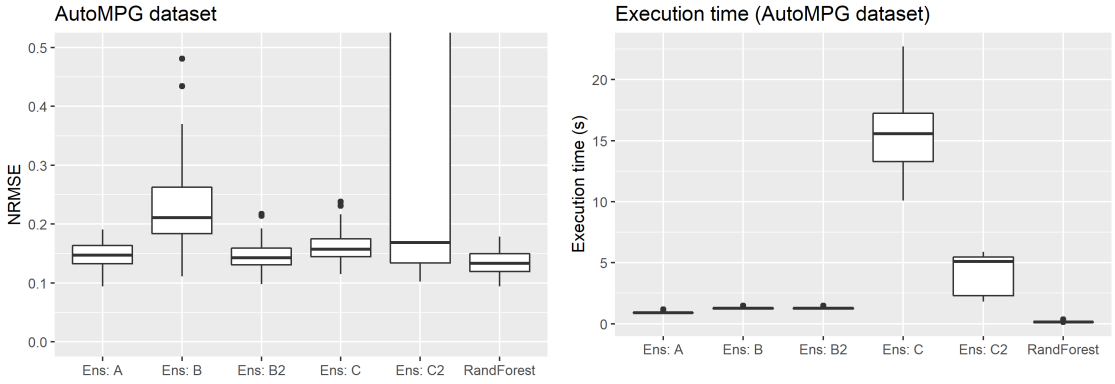

Figure 7.18: Results of experiment 2 for the AutoMPG dataset.

Communities dataset

Figure 7.19 shows the results of the Communities dataset. The performance metrics obtained for the tested methods are very similar, but again, the A and B2 seem to be the ensemble learners that better perform. In this case, the EnsSNN obtains similar results than the Random Forest. About the execution time, again the C and C2 are the worst cases, but now, with a dataset with more than 1000 observations, the execution time of method A is lower than the Random Forest.

MV dataset

Finally, a large regression dataset is tested. As done in experiment 1, the number of repetitions is reduced to 5 because the time for training a model is quite high. Also, the C and C2 methods are

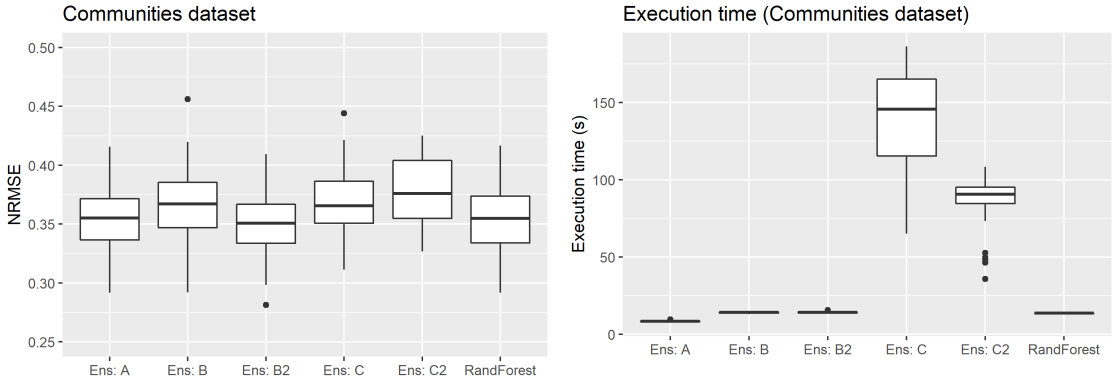

Figure 7.19: Results of experiment 2 for the Communities dataset.

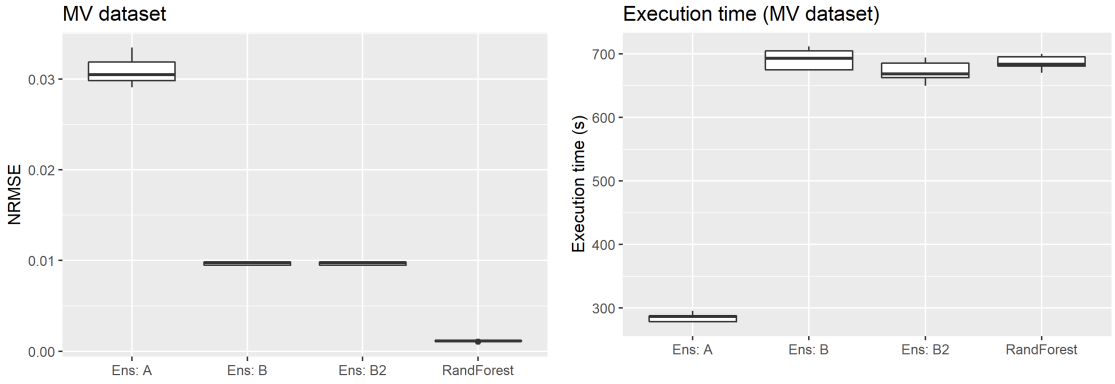

Figure 7.20: Results of experiment 2 for the MV dataset.

not executed for large datasets for the same reasons (execution time issues). Figure 7.20 shows the results of the MV dataset. These results show that the Random Forest is the learner that obtains better NRMSE, but the EnsSNN methods are very near (the differences of NRMSE is lower than 0.01). In this case, the B and B2 methods get better predictions than A. About the execution time, the A method is much faster than all others.

7.2.2 Binomial classification

Heart dataset

Let us see the behaviour for binomial classification problems. Figure 7.21 shows the results of the Heart dataset. For this dataset, A and A2 obtain the best accuracies among all methods (included the Random Forest). The B2 is a step below, but it is very near, and, interestingly, the B method is the worst one, it seems that this method fails by overfitting issues. About the execution time, it happens the same than with the regression problems, the C methods are the ones that need more time to train the model.

Horse Colic dataset (binomial case)

Figure 7.22 shows the results of the Horse Colic dataset. These results are similar to the Heart dataset, but in this case, the Random Forest performs a little better than the A and A2 methods. Also, the B2 obtains similar results than A and A2. All other methods (B, C and C2) predicts worse the response variable.

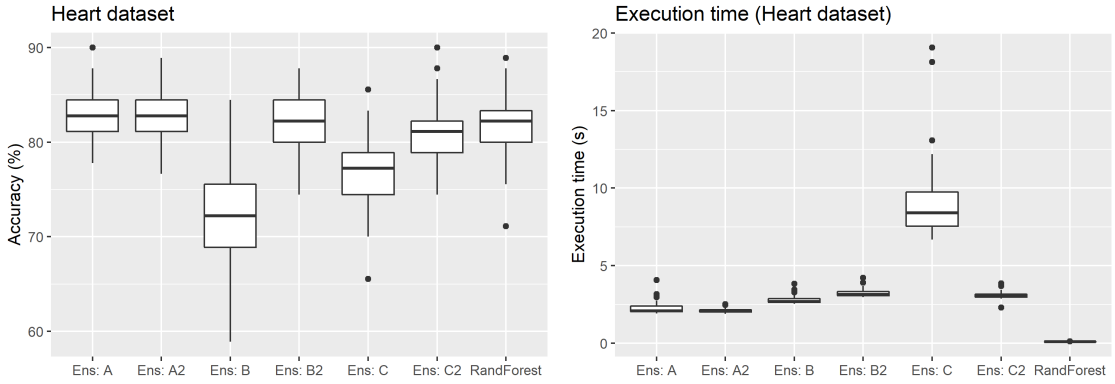

Figure 7.21: Results of experiment 2 for the Heart dataset.

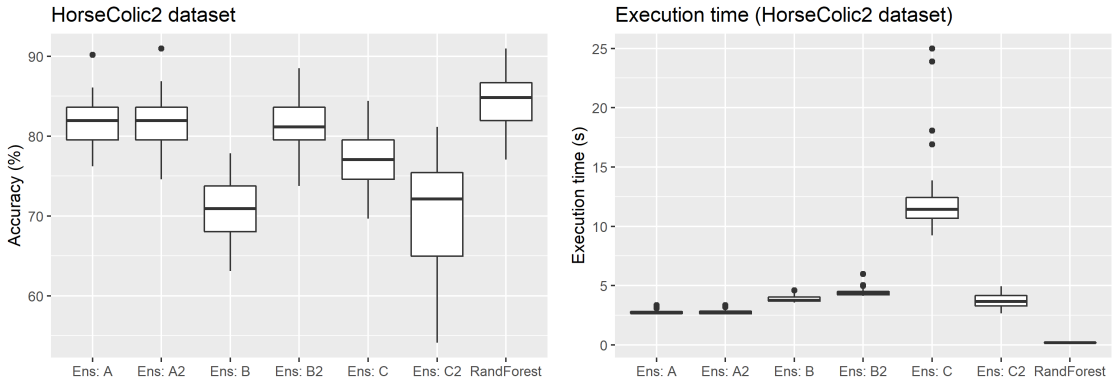

Figure 7.22: Results of experiment 2 for the Horse Colic dataset (binomial case).

Pima dataset

Figure 7.23 shows the results of the Pima dataset. Again, we see similar results than with the previous datasets. Method A, A2, B2 and Random Forest are the ones that best perform, with similar performance metrics, and the other methods are worse.

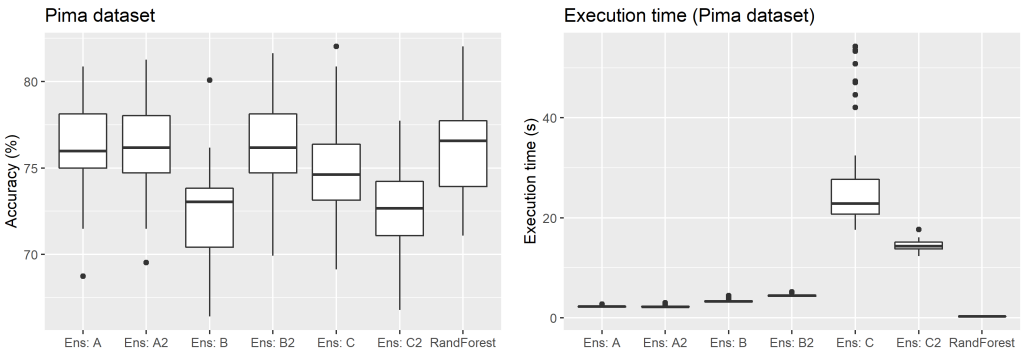

Figure 7.23: Results of experiment 2 for the Pima dataset.

Mammographic dataset

Figure 7.24 shows the results of the Mammographic dataset. A, A2 and B2 obtain similar accuracies, improving the results obtained by the Random Forest. It is also important to mention that the B2 needs more time for training the model than the A methods.

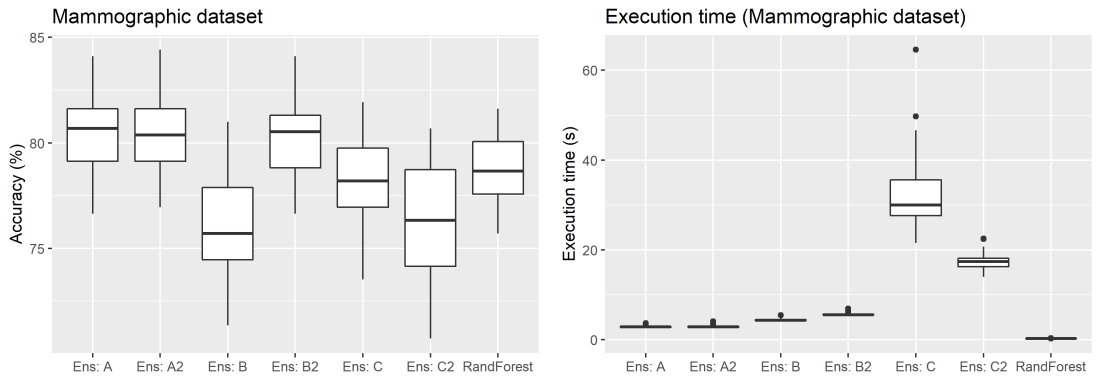

Figure 7.24: Results of experiment 2 for the Mammographic dataset.

Mushroom dataset

Figure 7.25 shows the results of the Mushroom dataset (C2 method is not shown in the figure because the accuracy was a lot worse than the other methods). For this case, all methods get high accuracy, but A, A2, B2 and Random Forest seem to be a step above and reach a nearly 100% of accuracy.

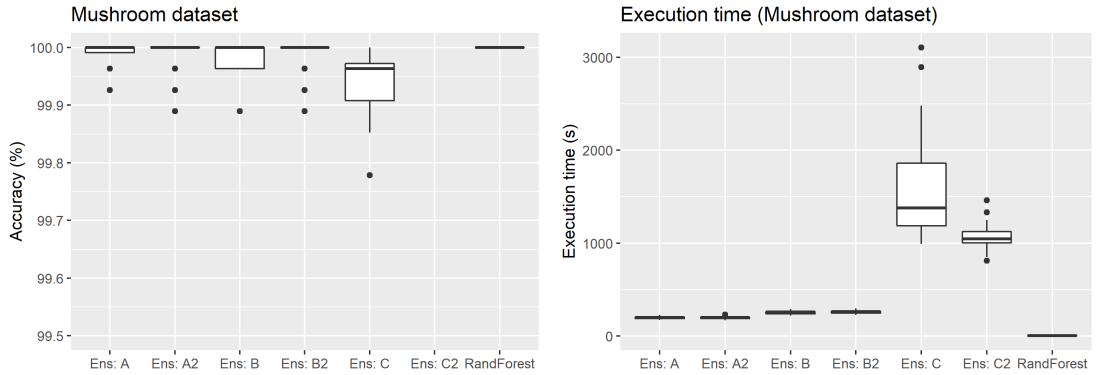

Figure 7.25: Results of experiment 2 for the Mushroom dataset.

Census dataset

Finally, it is tested with a large dataset (only 5 repetitions of the experiment are executed). For this dataset, the Random Forest gets the best performance metric, followed by B, A and A2. In

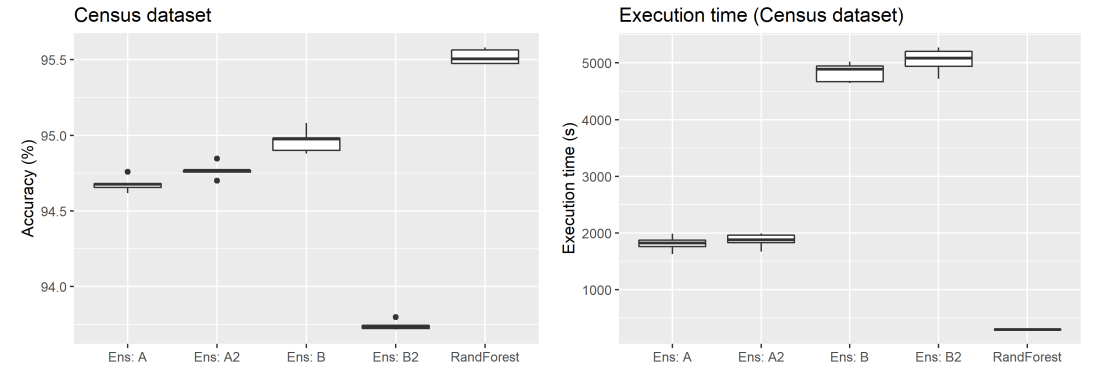

Figure 7.26: Results of experiment 2 for the Census dataset.

this case, adding regularization to B does not improve the accuracy.

7.2.3 Multinomial classification

Audiology dataset

Figure 7.27 shows the results of the Audiology dataset. Methods A, A2, B and Random Forest are the methods that best perform on the Audiology dataset, C and C2 are the worst cases, and regularizing the B method does not improve the accuracy of the model. About the execution time, it has a similar behaviour than the regression and binomial classification problems.

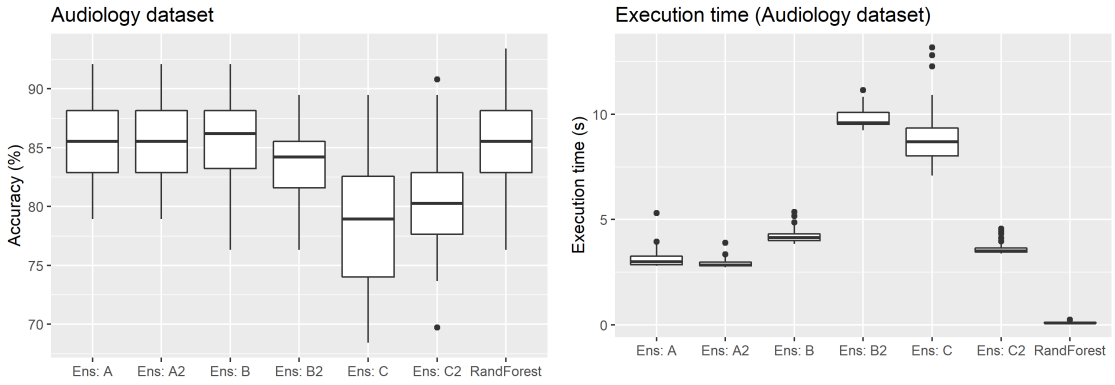

Figure 7.27: Results of experiment 2 for the Audiology dataset.

Glass dataset

For the Glass dataset, the Random Forest gets the highest accuracies, followed by the A, A2 and B2 methods that are very near. Again the Cs methods are the worse cases.

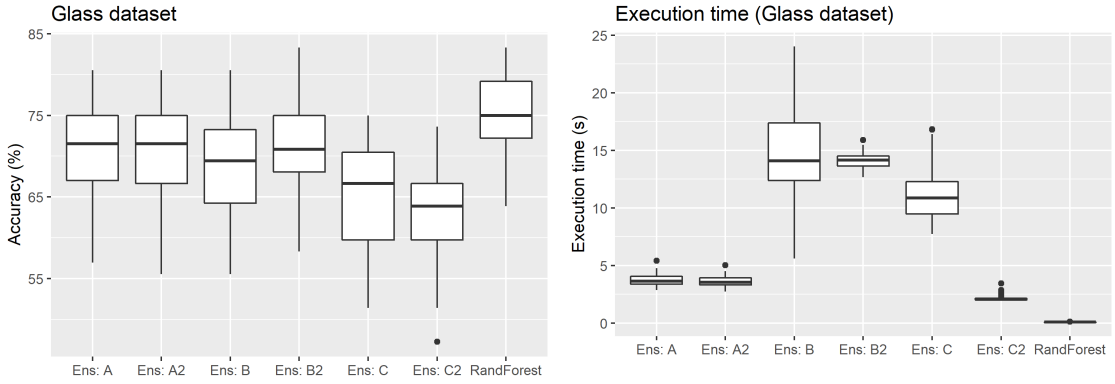

Figure 7.28: Results of experiment 2 for the Glass dataset.

Horse colic dataset (multinomial case)

The results of the Horse colic dataset are very similar than the ones obtained with the previous datasets: A, A2, B2 and Random Forest are the methods that better perform, having the Random Forest a slightly higher accuracy than the others.

Annealing dataset

The results of the Annealing dataset are quite different from the previous ones. First, the Random Forest is the worst method, even improved by C and C2. A and A2 are the best models and A2

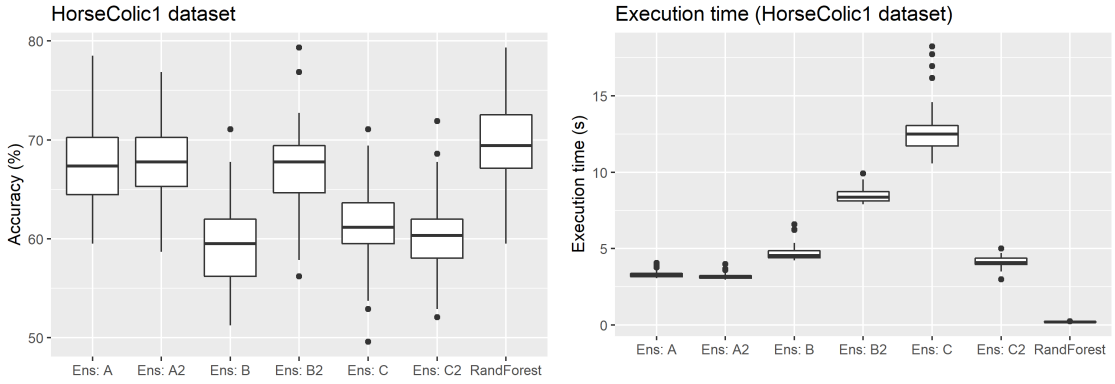

Figure 7.29: Results of experiment 2 for the Horse colic dataset (multinomial case).

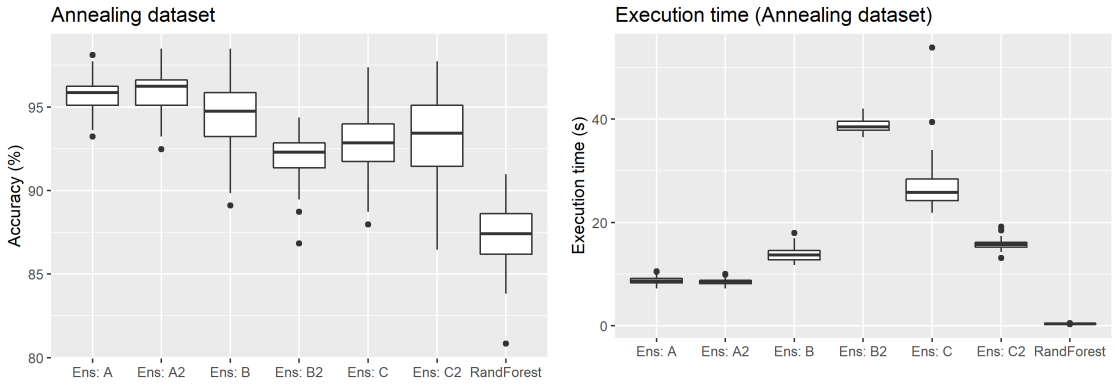

Figure 7.30: Results of experiment 2 for the Annealing dataset.

seems to be a step above. Also, for this dataset, the regularization of B does not improve the performance metrics.

Contraceptive dataset

Figure 7.31 shows the results of the Contraceptive dataset. A, A2 and B2 are the methods that obtain better accuracies for this dataset. These methods improve the predictions of the Random Forest. The behaviour among the other methods is the same than the previous datasets.

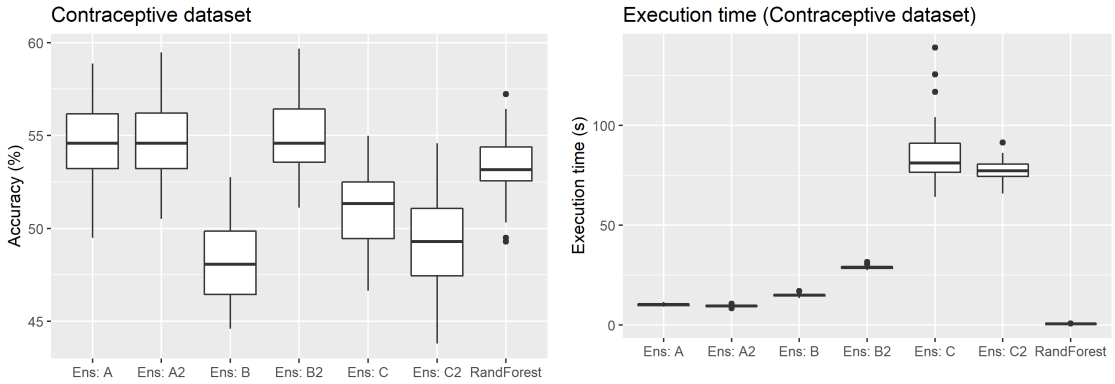

Figure 7.31: Results of experiment 2 for the Contraceptive dataset.

Diabetes dataset

Finally, the diabetes dataset is tested, a large problem. Only 5 repetitions are executed because each one could take hours. In Figure 7.32 we can see that Random Forest and B2 are the methods that get better predictions, followed by B, A2 and A (but these methods are very near, less than 0.5% of difference). About the execution time, the fastest methods are A and A2, that are quicker than the Random Forest. As expected, B2 takes more time to train the model than the others.

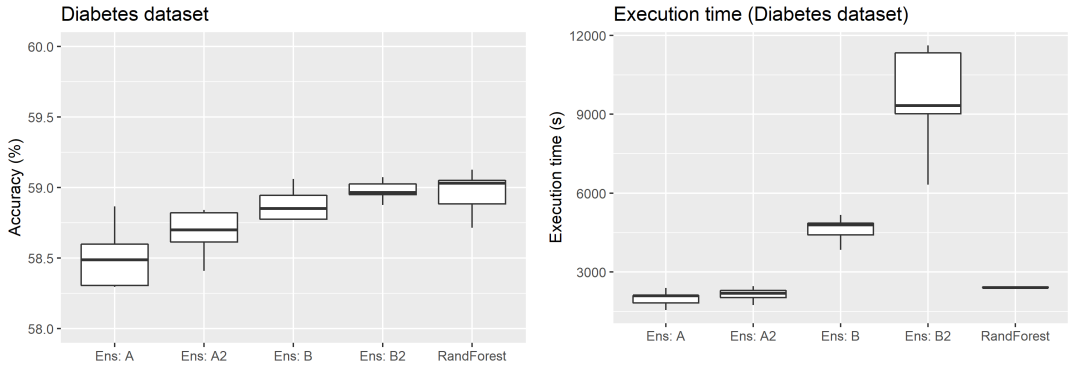

Figure 7.32: Results of experiment 2 for the Diabetes dataset.

7.2.4 Summary of Experiment 2

With the second experiment, I have analyzed the behaviour of the ensemble learners of the EnsSNN. The results of this experiment show that the EnsSNN gets performance metrics similar to the Random Forest (in some cases it is better and in some other cases it is worse), so the EnsSNN creates good models. Among the different ensemble learners, A and B2 are the best methods. But in some datasets, the regularization of B (B2) decreases the accuracy or NRMSE of the model. The C and C2 methods are a step below these methods and they perform worse in all datasets (performance metric and execution time).

About the execution time, C and C2 need too much time to train the model, they are followed by B2 and then by B, and finally, A and A2, that are the fastest ones. Also, I have noticed that the training time of the Random Forest is lower than the EnsSNN for small problems, but for large datasets, A and A2 are quicker than the Random Forest.

As a conclusion of this experiment, we have seen how the EnsSNN performs in a set of datasets and it equalizes the results of the Random Forest and, in some cases, it is improved. Moreover, the A ensemble methods are a step above to the other EnsSNN methods in terms of performance metric and especially in execution time.

7.3 Experiment 3

Analysis of the number of prototypes of a single SNN

For the first *complementary experiment*, all parameters of the model are fixed with the default values and it is played with the number of prototypes of a single SNN in order to see its behaviour. For this experiment and all following ones (*complementary experiments*), to reduce the number of figures it is shown a graphic with the mean Accuracy/NRMSE and mean execution time of the repetitions (for *complementary experiments*, 20 repetitions are executed).

In Appendix B.3 there are the numerical results of this experiment and Figure 7.33 shows them graphically. With these graphics, we can see that using a low percentage or a high percentage of prototypes makes the model worse, so the best results are found in the intermediate percentages. With very few prototypes the network cannot learn all the information needed to predict the response variable. In the other hand, when the network has a high percentage of observations that are prototypes, then the network fails by overfitting issues. I want to stress the fact that the

network does not use regularization, and maybe, if it is used regularization when the percentage of prototypes is high, the decrease in the performance metric could be lower.

In most datasets, there is an improvement on the performance metric when increasing the number of prototypes from 0 to 10%, then from 10% until 100% this performance metric remains stable or it decreases. Looking at these results a good choice for the percentage of prototypes would be to use 10% of the data as prototypes.

About the execution time, as expected, most of the datasets need more time to be trained when the number of prototypes is higher, so for this reason, the graphics show a positive slope (Contraceptive is an exception).

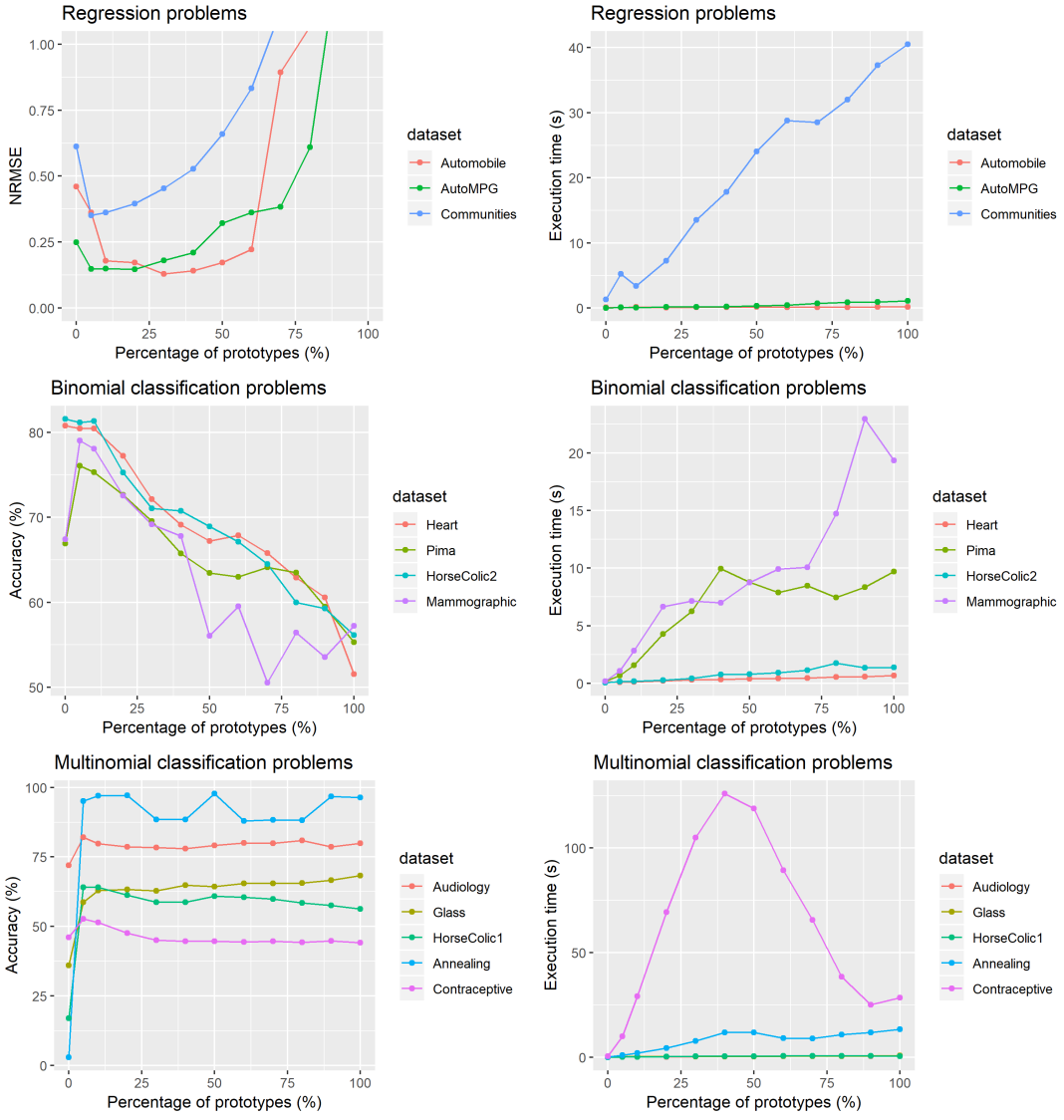

Figure 7.33: Results of experiment 3. From left to right: performance metric plot and execution time plot. From top to bottom: Regression, binomial and multinomial classification problems.

7.4 Experiment 4

Comparison of PAM and Random prototype selection for the Ensemble of SNNs

For this experiment, it is set all parameters of the Ensemble of SNNs to its default values and it is analyzed the prototype selection methods used in each SNN (PAM or random). In Appendix

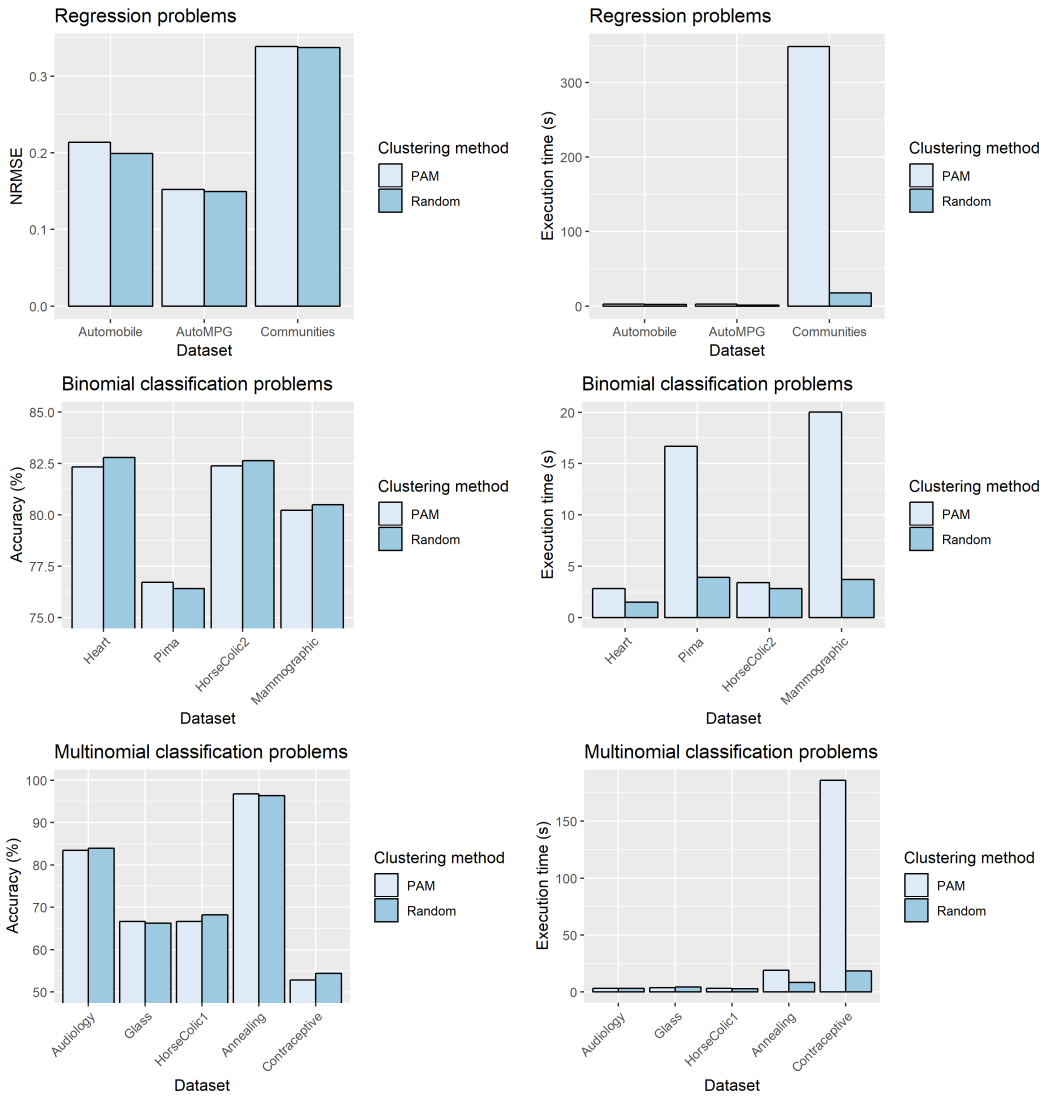

Figure 7.34: Results of Experiment 4. From left to right: performance metric plot and execution time plot. From top to bottom: Regression, binomial and multinomial classification problems.

B.4 there are the numerical results of this experiment and in Figure 7.34 the graphical results are shown.

The first thing that we see in these plots is that the PAM method spends more time in training the model than the Random prototype selection. This fact was expected because the random method is very simple and its computational cost is very low, oppositely to PAM, that needs to compute the similarities between all observations and create the clusters.

Secondly, with the performance metric plots, we see that there is not a huge difference among the two methods, but it seems that in most of the cases, the random method obtains slightly better results than the PAM. This may be caused because the ensembles usually have better performance when the learners are different among them, and training each SNN with the Random method instead of PAM will add more variability among learners.

7.5 Experiment 5

Analysis of adding regularization to each SNN of the Ensemble

It is analyzed the influence of adding regularization to each SNN of the Ensemble. To do that, all other parameters of the Ensemble are set to its default values and it is played with these regularizations. Appendix B.5 contains all numerical results of the experiment and in Figure 7.35 we can see them graphically.

When the SNNs do not use regularization, in most of the problems, there is an improvement of the NRMSE/Accuracy (except for two binomial classification problems), this improvement is not very high but it is significative. Ensembles (as a general method) usually perform better when some of its learners overfit the data. So, the behaviour of the regularization of the SNNs may be caused by that, the overfitting of each SNN will be higher when there is no regularization in the SNNs. About the execution time, these regularizations increase the time to train the model because it needs to find the regularization hyper-parameter λ (for the regression problems the difference is lower because it does a GCV instead of a 10-fold-CV).

Summarizing, adding regularization to each SNN of the ensemble does not improve the predictions and it spends more time training the model.

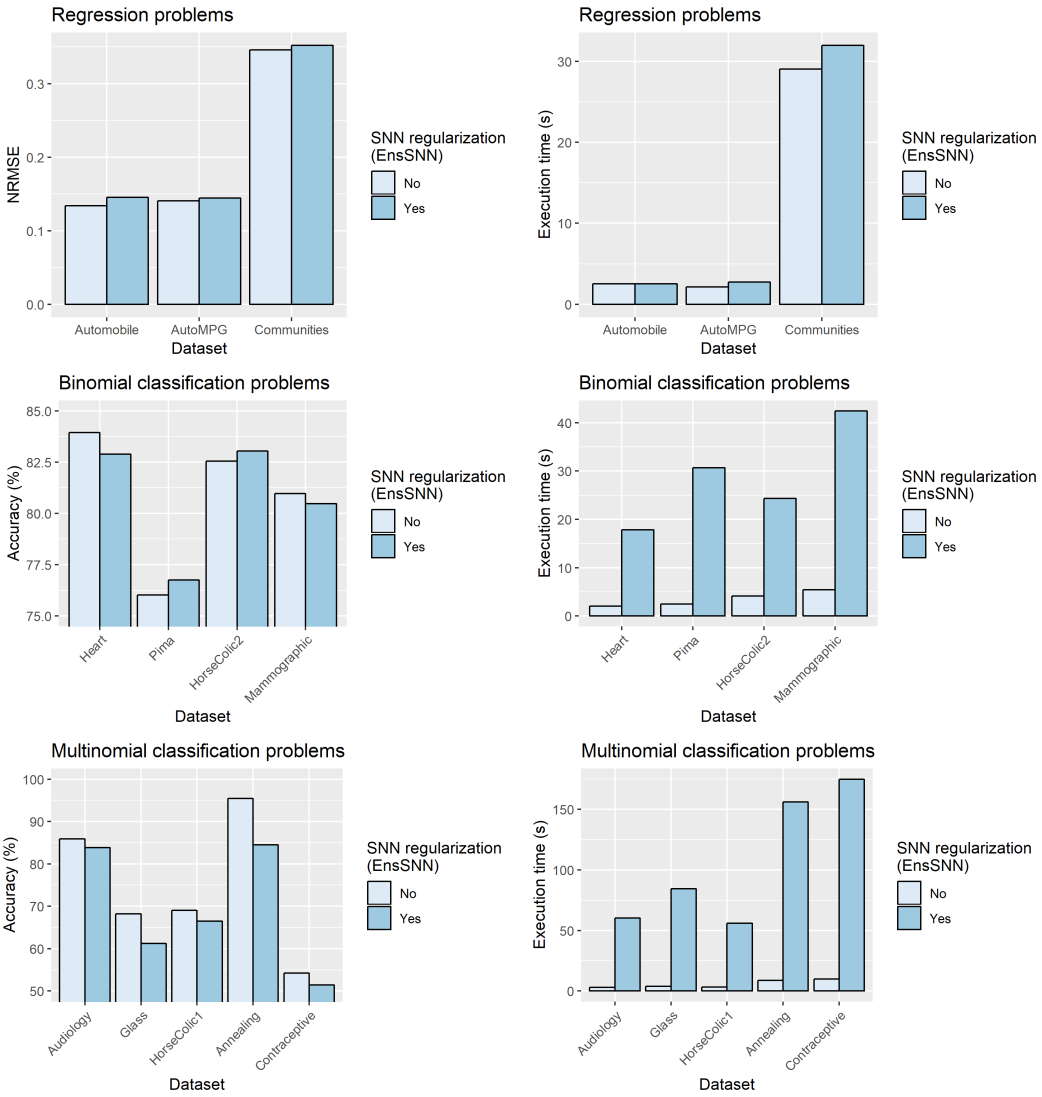

Figure 7.35: Results of experiment 5. From left to right: performance metric plot and execution time plot. From top to bottom: Regression, binomial and multinomial classification problems.

7.6 Experiment 6

Analysis of the distribution used to select the number of prototypes of each SNN of an Ensemble of SNNs

It is analyzed the influence of the way to choose the number of prototypes of each SNN of the Ensemble. All other parameters of the Ensemble are set to the default values and it is set that all distributions have the same expected value. Appendix B.6 contains all numerical results of the experiment and Figure 7.36 shows them graphically.

In these numerical and graphical results, there is not a significant difference among the methods. All distributions seem to obtain similar Accuracy/NRMSE and they spend the same time training the models. So, this feature seems to do not have an important influence on the performance metrics.

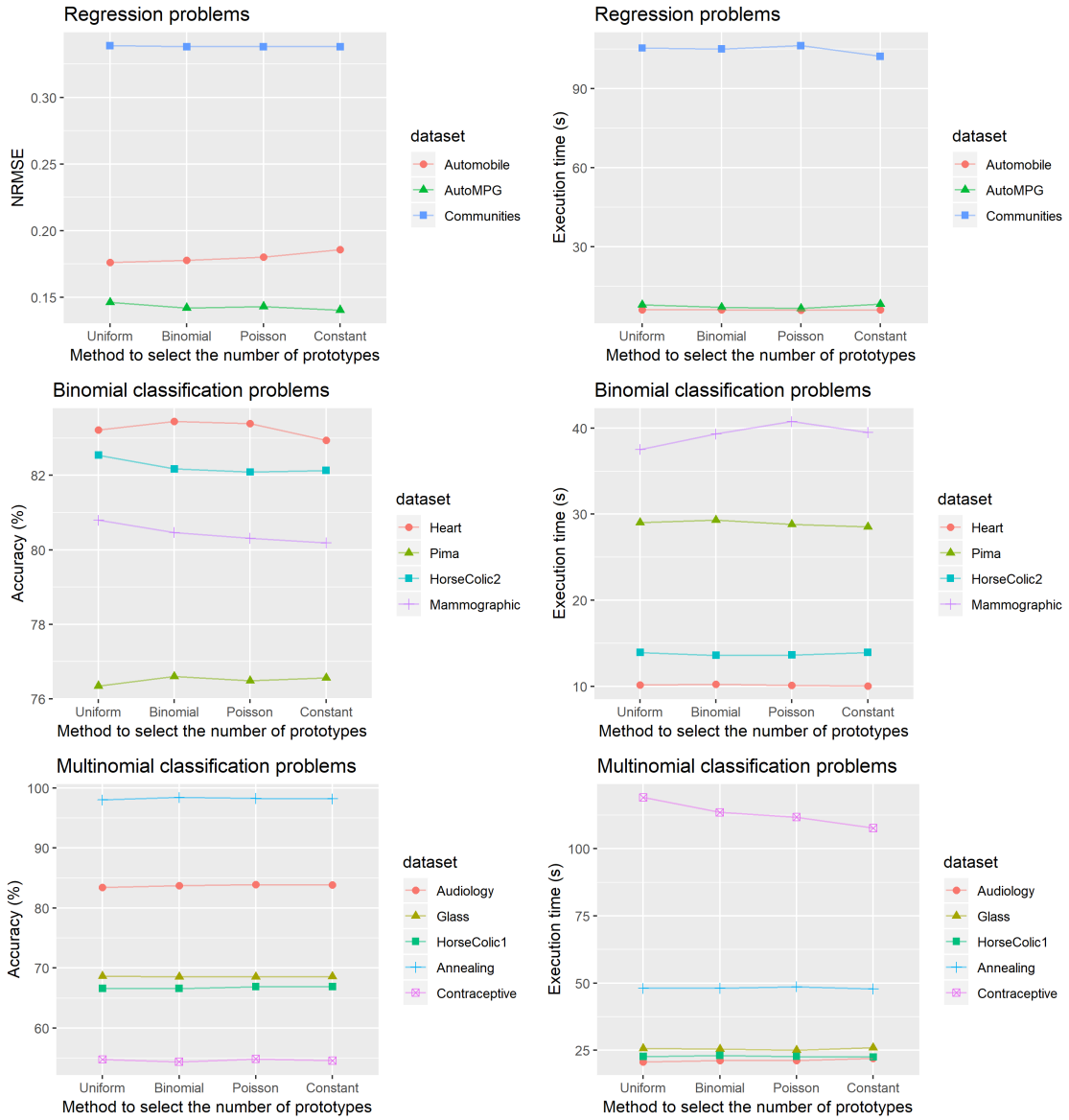

Figure 7.36: Results of experiment 6. From left to right: From left to right: performance metric plot and execution time plot. From top to bottom: Regression, binomial and multinomial classification problems.

7.7 Experiment 7

Analysis of the method used to select the number of observations to train each SNN of an Ensemble

With this experiment, it is analyzed the influence on the performance metric that has the distribution that chooses the number of observations used to train each SNN of the Ensemble. It is set the expected value of these distributions to be the 50% of the number of observations (all other parameters are set to the default values). Appendix B.7 contains all numerical results of the experiment and Figure 7.37 shows them graphically.

As with the previous experiment, there is not a significant difference between using one distribution or another. So, it seems that the way to select the number of observations does not have an important influence on the performance metrics.

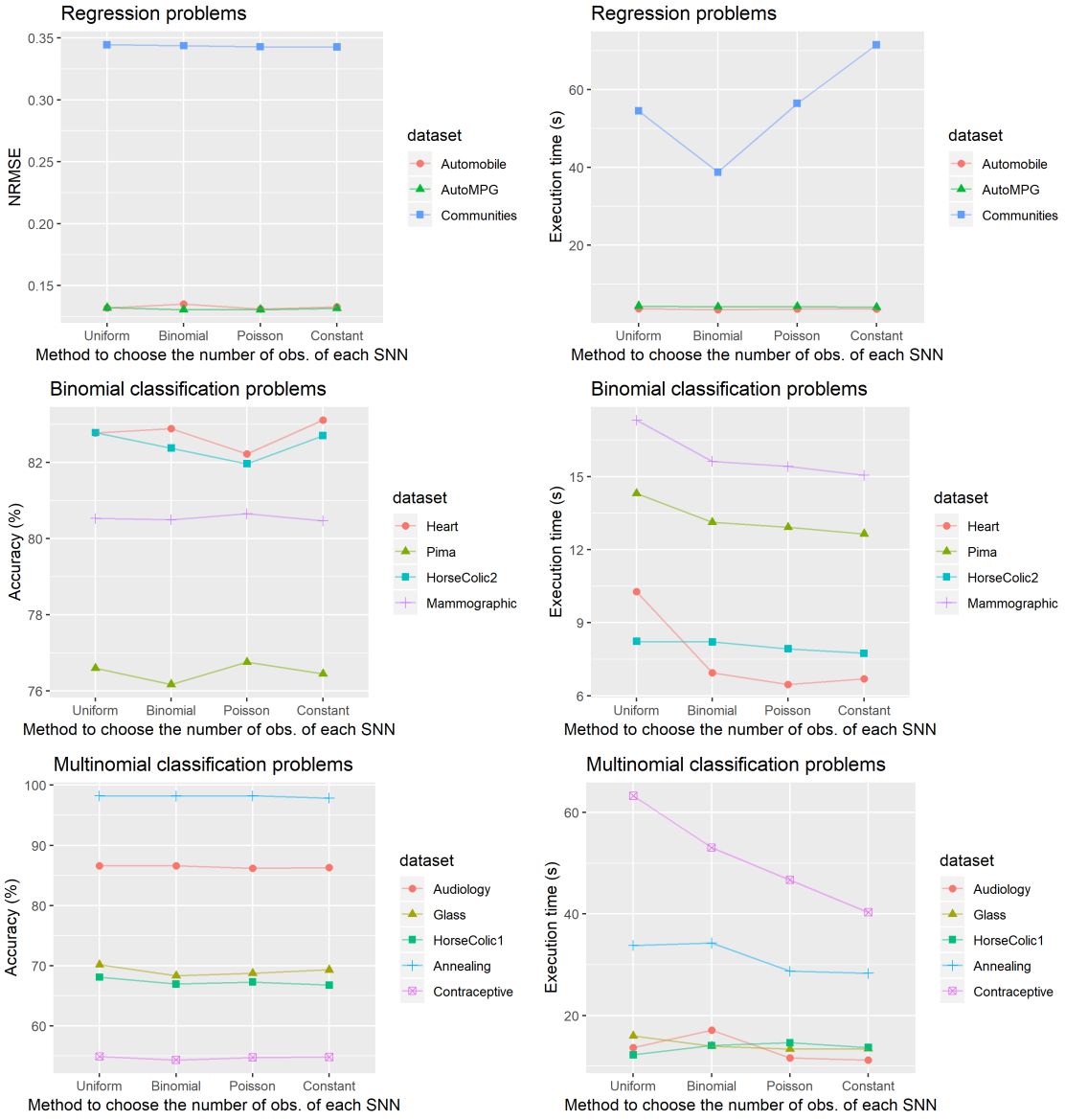

Figure 7.37: Results of experiment 7. From left to right: performance metric plot and execution time plot. From top to bottom: Regression, binomial and multinomial classification problems.

7.8 Experiment 8

Analysis of the proportion of data used to train each SNN of an Ensemble

With the eighth experiment, it is analyzed the number of observations used to train each SNN of an Ensemble. The constant distribution is used and all other parameters are set to the default values. Appendix B.8 contains all numerical results of the experiment and Figure 7.38 shows them graphically.

With these figures, we see that when the percentage of observations is very low the ensemble predicts very bad, and when the SNNs are trained with more observations, then the performance metric increases. But of course, using more observations will increase the learning time of each SNN, so, we should find a threshold between the two criteria. For these problems, I think that with 50% of the data the Accuracy/NRMSE is almost the maximum value and the training time is not too high. Also, it is important to mention that for large problems, it may be convenient to reduce this percentage in order to speed up the training time of the model.

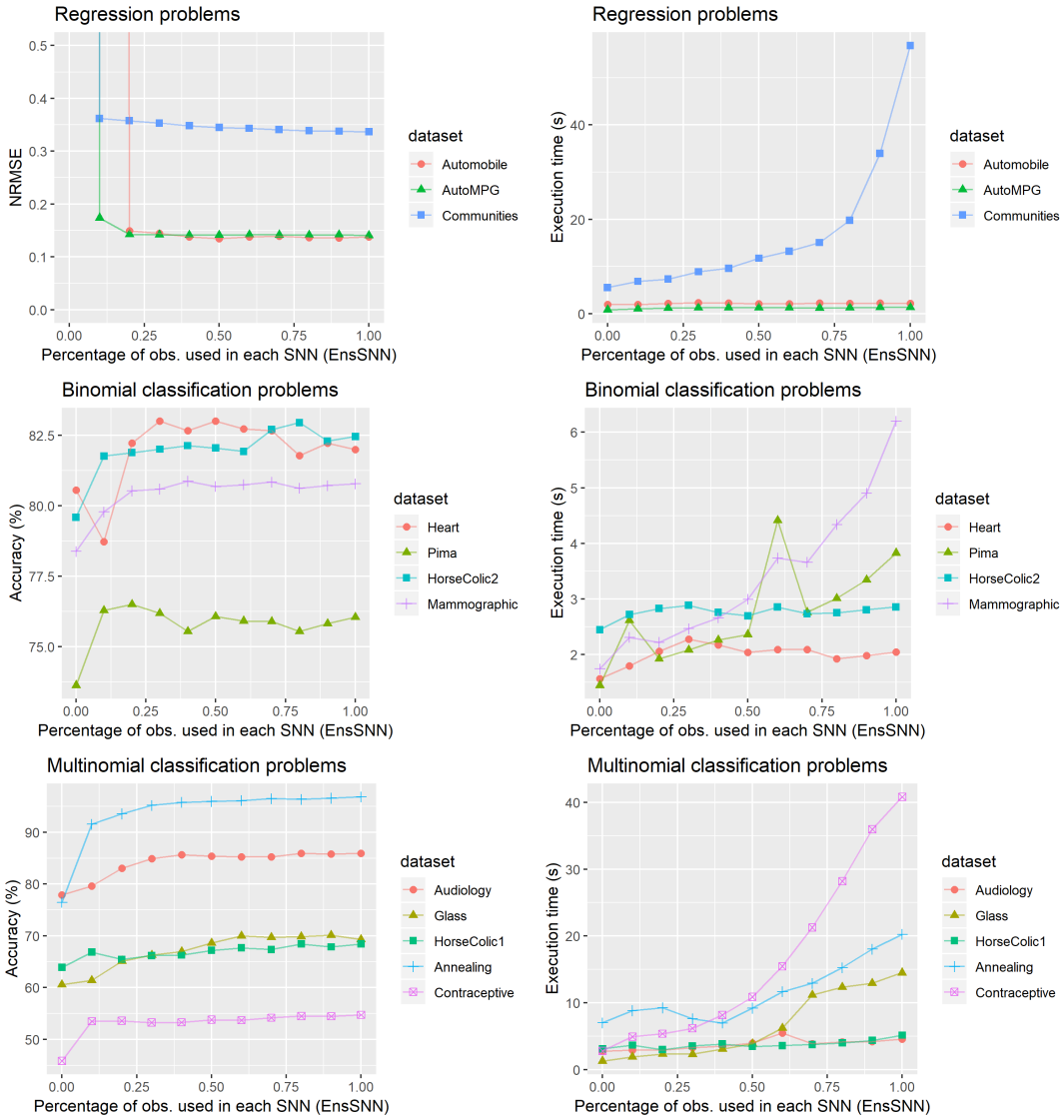

Figure 7.38: Results of experiment 8. From left to right: performance metric plot and execution time plot. From top to bottom: Regression, binomial and multinomial classification problems.

7.9 Experiment 9

Missing values analysis (Ensemble of SNNs)

Finally, the last experiment analyzes how robust is the Ensemble of SNNs against the missing values. To do that, it is randomly set to *NA* a certain percentage of the total training data. Appendix B.9 contains all numerical results of this experiment and Figure 7.39 shows them graphically.

As expected, as more missing values are in our dataset, worse is the performance of our model (Accuracy/NRMSE) because more information is removed from the data. Each dataset has a different behaviour, but with 50% of missing data, most of the models still give a proper prediction. Moreover, the percentage of missing values seems to do not influence on the execution time.

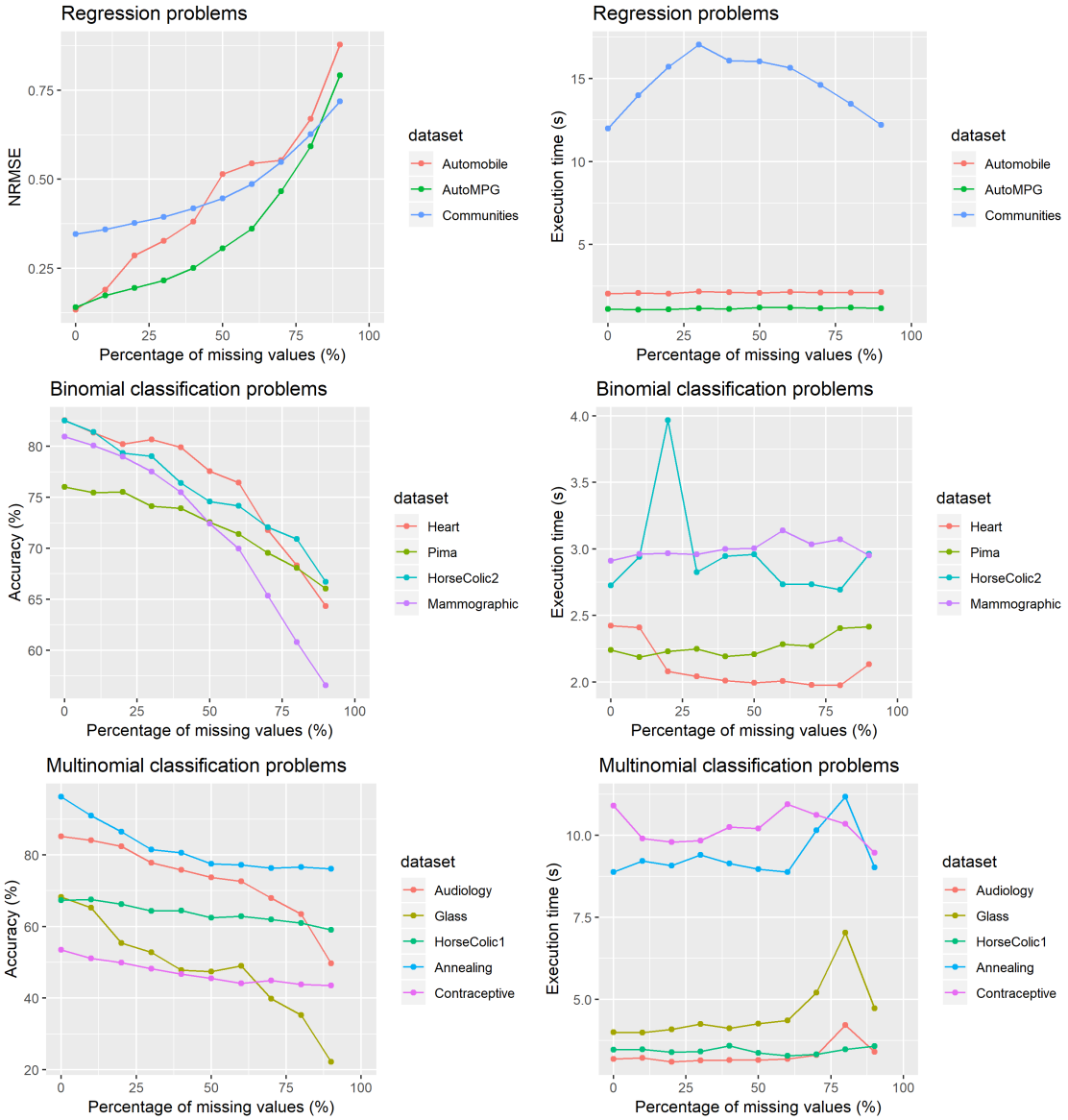

Figure 7.39: Results of experiment 9. From left to right: performance metric plot and execution time plot. From top to bottom: Regression, binomial and multinomial classification problems.

7.10 Final discussion of the experiments

With these nine experiments, we have tested and we have seen the behaviour of the SNN and the Ensemble of SNNs for a set of problems (small and large). With the *main experiments*, the main variants of these learning methods are tested, and with the *complementary* ones, it is tested specific features of them.

With the first experiment, we have seen that the SNN improves the accuracy/NRMSE of the classical decision tree in almost all cases. Also, the regularization of the linear model has a high impact on the performance of the model (in most of the cases, it improves it). Then, the optimization of p has some influence but much lower than the regularization. And finally, the way to choose the prototypes has low influence.

With the second experiment, we have seen that the Ensemble of SNNs gets a performance metric similar to the Random Forest and, in some cases, it improves it. About the ensemble learner, the A and A2 are the two methods that best perform, in almost all cases they are the methods with the best accuracy/NRMSE and, moreover, they are the fastest in training the model.

In these two experiments, the methods are tested with small and large datasets, and with both sizes, the results obtained are quite good (similar to Random Forest). It is true that for large problems it takes time (hours) to train the model, but the created models get good performance metrics.

The missing part is to compare the results of the SNN and the Ensemble of SNNs. In Appendix B.1 and B.2 are the numerical results of these experiments (where we can find the mean and variance of Accuracy/NRMSE). The results of the two learners are very similar, but the Ensemble of SNNs seems to obtain slightly better results than the SNN. Table 7.1 shows a summary with the configurations that better perform of each method: for the Ensemble, the A learner method is shown, and for the SNN it is shown the random prototype selection, p found by cross-validation and regularization (it is also shown p constant because the CV was not executed for large problems). With this table, we can see that, among the thesis proposals, the Ensemble of SNNs is the best in many of these problems (best means having the highest accuracy or lowest NRMSE), and when it is not the best, it obtains performance metrics very near to the SNNs. Despite this, the SNNs obtain quite good results too. Moreover, it is important to mention that the Ensemble is faster when the problem is large because it trains n small SNNs instead of one with all the data (that would take more training time). So, for large problems, even larger than the ones tested in this

	Ensemble of SNNs Method A (Accuracy/NRMSE)	SNN Rand , p=Const, Reg (Accuracy/NRMSE)	SNN Rand , p=CV, Reg (Accuracy/NRMSE)	Random Forest
Automobile	**0.1349**	0.1902	0.1608	0.0843
AutoMPG	0.1488	0.1647	**0.1409**	0.1344
Communities	**0.3530**	0.3545	0.3539	0.3540
MV	0.0309	**0.0039**	-	0.0011
Heart	**83.11**	82.88	82.68	81.62
Pima	**76.42**	76.20	76.34	76.13
HorseColic2	81.86	81.72	**82.03**	84.45
Mammographic	80.41	80.39	**80.62**	78.74
Mushroom	**99.98**	96.53	-	100
Census	94.67	**94.70**	-	95.51
Audiology	**85.68**	81.23	81.86	85.23
Glass	**70.97**	46.72	48.97	75.41
HorseColic1	**67.50**	66.66	66.79	69.75
Annealing	**95.72**	90.86	91.11	87.50
Contraceptive	**54.49**	53.22	54.25	53.29
Diabetes	58.51	**58.87**	-	58.96

Table 7.1: Comparison of the results of SNN and Ensemble of SNNs. Also, the results of the Random Forest are shown as a benchmark.

thesis, the Ensemble with method A will be quicker than a simple SNN. Another thing to comment among the two learners is that the Ensemble of SNNs, in most of the problems, gets lower variance of Accuracy/NRMSE than the SNN. So, the variability between two different executions of the learning method is higher in the SNN case (one of the reasons to use an ensemble was to reduce this issue). Also with the Table 7.1, we can see that our proposals get similar results than the Random Forest (in some cases they are better than Random Forest and in some other cases they are worse).

Finally, with the *complementary experiments*, we have understood the behaviour of some of the features related to these methods. For example, it was seen how robust the Ensembles of SNNs are against missing values or the influence of the number of prototypes of a network.

Chapter 8

Conclusions

In this chapter, the main conclusions of the thesis are presented and it is reviewed if the initial objectives were fulfilled. In this document, I have designed and implemented two learning methods: SNN and Ensemble of SNNs.

The SNN is a two-layer neural network that is based on similarity measures. In the design section, several variants of this network are explained, and with the experiments, I have seen the influence of each one of them. The results suggest that adding regularization to the SNN is the feature with more influence among the tested ones. In most of the problems, this regularization of the network makes an increase in the performance metric of the model (accuracy or NRMSE). Another feature that was tested but it has less impact than the regularization is the value of p of the activation function. Several methods were designed and the one that performs better was the cross-validation and the one that performs worst was the constant value. Also, an optimization procedure was designed to set the value of p, but as it does not allow regularization, the results are worse than any other p method with regularization (this could be a future work to be done). Finally, it is also analyzed if selecting the prototypes of the network by a clustering procedure is better than do it randomly, and the results of the experiments suggest that there is not a high impact on choosing one method or another. Moreover, the results of the SNN improves the Accuracy/NRMSE obtained with the decision tree in almost all experiments.

Secondly, I have designed the Ensemble of SNNs. Several ensemble learners were proposed, and, in most of the cases, the best results were obtained when it was used the A and A2 methods. These methods are the simplest ones, they do the mean and the majority vote of the learners, so they are the methods that get the best accuracy/NRMSE and also the fastest ones. More complex methods were also tried but none of them beat the A methods. Method B proposes to apply a GLM with the responses of the learners. In some cases, the results of B were near to the As methods but in others, they fail by overfitting issues. Also adding regularization to the GLM (Method B2) gets better results than B, but these results seem to be a step below than method A in terms of Accuracy/NRMSE and execution time. And finally, a Mixture of Expert approach is also proposed (Methods C and C2), but the results of the experiments show that it is the worst method: it gets very bad performance metrics compared to the other methods, and also it is the one that spends more time in training the model (this approach does not seem to work well for the SNNs). The Ensemble of SNNs and all its variants were compared with the Random Forest. Method A got similar accuracy/NRMSE than the Random Forest, and for some of the problems, it obtained better results.

Finally, there is a comparison between the two learning methods proposed in this thesis. In most of the cases, the Ensemble gets slightly higher performance metric than a simple SNN (but the SNN is very near). Moreover, the Ensemble usually gets a lower variance of Accuracy/NRMSE too. So, designing and using the Ensemble was a good idea to improve the SNN. About the execution time, the ensemble takes more time when the problem is small, but when the problem is large, the complexity of the Ensemble is lower than the SNN because it has to fit M *small* SNNs instead of one *large* SNN. So the results of the experiments suggest that for large problems the Ensemble should be preferable.

Also, one of the goals of this thesis was to understand and know if the use of similarity measures

in the learning algorithm would create good models compared to classical methods. Against the decision tree, the experiments show that the SNNs (and Ensemble) get better performance metrics than the decision tree in almost all problems. When they are compared to the Random Forest, the results are closer: in some cases, the Random Forest gets better performance metrics, and in some others, the Ensemble of SNNs gets better results. I want to stress the fact that the Random Forest is a learning algorithm that is difficult to beat because its degree of randomness among the learners is very well chosen. So, the results obtained with the Ensemble of SNNs are quite good and competitive. But there is further scope for improvement, one way to improve could be to analyze and refine the randomness of the SNNs of the ensemble.

Summarizing, in this thesis, I have researched and designed two new learning methods, and the results of the experiments show a competitive performance of both methods, obtaining similar results than the Random Forest and improving it in some problems.

Review of the thesis objectives

At the beginning of this document, there is a list of the main objectives of this thesis. Let us review if they have been accomplished:

1. *Design a Similarity Neural Network (SNN).*

✓ In Section 4.1, the design of the SNN with all its configurations is explained.

2. *Implementation of the SNN.*

✓ I have implemented an *R* function that creates an SNN and another one that predicts new data (Section 5). I have tried to encapsulate all the functionality in these functions and make them be user-friendly functions that could be used for everyone.

3. *Add regularization to the SNN.*

✓ It is added regularization to the GLM of the network, and the results of the experiments show an improvement on the Accuracy/NRMSE.

4. *Optimize the p hyper-parameter of the activation function of the hidden neurons.*

✓ As said before, it is designed a set of methods to choose the p values of the activation function. The cross-validation method seems to perform better than when using a constant value. Also, there is the optimization procedure, but it fails in the way that it does not allow regularization, and regularization has a high impact on the performance metrics.

5. *Design an Ensemble of SNNs*

✓ In Section 4.2, the design of the Ensemble with all its configurations is explained.

6. *Implement the Ensemble of SNNs*

✓ As done with the SNN, I have implemented an R function that creates an Ensemble and another one that predicts new data (Section 5).

7. *Test the performance of the SNN and Ensemble of SNNs by designing experiments.*

✓ Nine different experiments are designed to test how these learning methods perform, and they were tested with 16 different problems (Section 6).

8. *Comparison between the Similarity Neural Network and the methods in the literature*

✓ The *main experiments* compare the learning methods with the decision tree and Random Forest. For most of the problems, the performance metrics are very similar to the Random Forest, and in some cases, they improve it.

9. *Test the performance of the learners with large problems.*

✓ Three of the datasets tested in the experiments are large problems, so they have been tested and the results are quite good.

Chapter 9

Future work

Finally, a list of tasks is proposed that would continue with the work done in this document. As said in previous chapters, the Similarity Neural Network (SNN) is an area that is less researched than other methods, it would need more research.

Some of these future works that would continue and extend this thesis are:

- First of all, it could be designed and executed more experiments with more datasets. These experiments could test the SNN and the Ensemble in more different conditions than the ones tested in this document.
- Also, the SNNs and the Ensembles could be tested with datasets that are much larger than the ones tested in this thesis, and we could see how these methods perform on very large problems.
- The SNN could be extended to allow more similarity measures / data types (e.g fuzzy or cyclic).
- The Gower's similarity has a set of parameters that can tune the weight of each feature of the dataset (w_k). These weights could be optimized for the predictive task of the problem.
- The last layer of the SNN is a linear model (GLM) of the similarities given by the hidden neurons. It could be tried to use another learning method, for example, an SVM or a MLP-NN instead of the GLM.
- The optimization procedure that sets the value of p (activation function) does not allow regularization. So, an interesting future work would be to add this regularization to the method and see how the model performs.
- Also, the regularization could be added to method C of the Ensemble of SNNs. It could be interesting to see if, with this regularization, the performance metric of the model improves.
- For the Ensemble of SNNs, it could be analyzed how decorated and distinct are the SNNs of the Ensemble. As said before, the ensemble works better when the learners are different among them and they disagree and do different predictions. So, one future work could be to analyze how correlated are the SNNs of the Ensemble. Then, if the degree of correlation is high, it could be tried some of the techniques to decorate and add randomness among SNNs. For example, it could be done a feature selection approach to the ensemble.
- In this thesis, it was designed a Bagging-like ensemble method. Another future work would be to design another type of ensemble, for example, a Boosting-like ensemble.
- The learning methods proposed in this thesis were implemented in the R language. This code could be refined and added to a package in order to be available for the community.
- Also related to the implementation, it could be optimized the code to make it faster. Moreover, it can be implemented in another language (e.g. C) in order to speed up the training time.

- Theoretically, the SNN can deal with semi-supervised datasets (observations without a label can be prototypes). The implementation can be modified to allow this type of datasets and it can be tested its behaviour with real problems.

These are some of the possible future works of this thesis, but as said before, there is a lot of work that can be done in the research area of similarity-based learning methods for heterogeneous data.

Acronyms

ANN Artificial Neural Network.

EnsSNN Ensemble of SNNs.

GCV Generalized Cross-Validation.

GD Gradient Descent.

GLM Generalized Linear Model.

LOO-CV Leave-One-Out Cross-Validation.

MLP-NN Multilayer Perceptron Neural Network.

MoE Mixture of Experts.

MSE Mean Squared Error.

NA Not Available.

NRMSE Normalized Root Mean Square Error.

PAM Partitioning Around Medoids.

RBF-NN Radial Basis Function Neural Network.

SD Standard Deviation.

SNN Similarity Neural Network.

SSE Sum of Squares Error.

SVM Support Vector Machine.

Lists of Symbols

x_i: Explanatory variables (input variables or features) of the ith observation of the dataset.

x_{ij}: Value of feature j for the ith observation.

y_i: Response variable of the ith observation of the dataset.

$\hat{y}_i$: Prediction of the response variable for the ith observation of the dataset.

N: Number of observations of the problems.

P: Number of explanatory variables.

K: Number of modalities of the response variable (multinomial case).

p: Parameter of the activation function f_p.

H: Number of prototypes or number of hidden neurons of an SNN.

hp: Expected number of prototypes for an SNN.

c_i: Prototype of the ith hidden neuron.

α: Weights of the linear model (SNN).

M: Number of SNNs of an Ensemble.

β: Weights of the ensemble learner.

Φ: Matrix of weights used in the gating network (Ensemble of SNNs).

Bibliography

[1] Lluís Belanche. Learning in networks of similarity processing neurons. *Workshop New Challenges in Neural Computation 2013*, pages 97–105, 2013.

[2] Lluís Belanche and Jerónimo Hernández. Similarity networks for classification: a case study in the Horse Colic problem. *Technical report, LSI-14-4-R*, 2014.

[3] Lluís Belanche and Jerónimo Hernández. Similarity networks for heterogeneous data. *ESANN 2012: the 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, Belgium from 25 to 27 April 2012: proceedings*, pages 215–220, 2012.

[4] Robert P.W. Duin, Marco Loog, Elzbieta Pękalska, and David M.J. Tax. Feature-based dissimilarity space classification. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6388 LNCS, pages 46–55, 2010.

[5] Robert P.W. Duin and Elzbieta Pękalska. The dissimilarity representation for structural pattern recognition. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7042 LNCS, pages 1–24, 2011.

[6] Jorge Orozco Luquero. Similarity and dissimilarity concepts in machine learning. *Technical report, LSI-04-9-R*, 2004.

[7] J. C. Gower. A General Coefficient of Similarity and Some of Its Properties. *Biometrics*, 27(4):857, dec 1971.

[8] Ethem Alpaydin. Multilayer Perceptrons. In *Introduction to Machine Learning*, chapter 11, pages 233–274. Second edition, 2010.

[9] Ethem Alpaydin. Radial Basis Functions. In *Introduction to Machine Learning*, chapter 12.3, pages 288–293. Second edition, 2010.

[10] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[11] Kevin P Murphy. Generalized linear models and the exponential family. In *Machine Learning: A Probabilistic Perspective*, pages 281–306. 2012.

[12] Stephen Boyd and Lieven Vandenberghe. Unconstrained minimization. In *Convex Optimization*, chapter 9, pages 457–512. 2004.

[13] Lluís Belanche and Julio Valdés. Similarity-based Heterogeneous Neural Networks. *Engineering Letters*, 14, 2007.

[14] Leonard. Kaufman and Peter J. Rousseeuw. *Finding groups in data : an introduction to cluster analysis - Partitioning Around Medoids (Program PAM)*. Wiley, 1990.

[15] Christopher M. Bishop. Mixtures of experts. In *Neural Networks For Pattern Recognition*, chapter 9.7, pages 369–371. 1995.

[16] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, oct 2001.

[17] Elzbieta Pękalska, Robert P. W. Duin, and Pavel Paclík. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, 39:189–208, 2006.

[18] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

[19] Christopher M. Bishop. *Neural Networks For Pattern Recognition*. 1995.

[20] Elzbieta Pękalska. *Dissimilarity representations in pattern recognition*. PhD thesis, 2005.

Appendix A

Optimization functions

A.1 Optimization of p: Error function and its derivatives

In Section 4.1.2.2, it is explained the methods used to decide the value of p (f_p) of an SNN. There is one that does the optimization of p and αs at the same time using the gradient descent algorithm. So, we should provide the objective function and its derivatives. In this section, it is explained and formulated in details this error function (the one that should be minimized) and its derivatives. Each type of problem (regression, binomial and multinomial classification) has a different error function, so each one should be treated separately.

A.1.1 Regression

For regression problems, it is used the SSE as the error function. It is defined N as the number of observations, H as the number of hidden neurons (prototypes), x_n is the vector of features for observation n, y_n is the response variable for observation n, s_g is a function that computes the Gower's similarity between two observations and c_h is the prototype of the hidden neuron h. Then, the error function (E) has two parameters: p and αs (vector of length $H+1$). So, the error function is defined as follows:

$$E(p,\alpha) = \sum_{n=1}^{N} (y_n - \hat{y}(p,\alpha,x_n))^2$$

$$\hat{y}(p,\alpha,x) = \sum_{h=1}^{H} \alpha_h f_p\left(s_g(x,c_h)\right) + \alpha_0$$

$$f_p(x) = \begin{cases} \frac{-p}{(x-0.5)-a(p)} - a(p) & \text{if } x \leq 0.5 \\ \frac{-p}{(x-0.5)+a(p)} + a(p) + 1 & \text{if } x \geq 0.5 \end{cases}$$

$$a(p) = -2^{-2} + \sqrt{2^{-4}+p}$$

Derivatives

The partial derivative of the previous error function with respect to p is:

$$\frac{\partial E(p,\alpha)}{\partial p} = -2\sum_{n=1}^{N}\left((y_n - \hat{y}(p,\alpha,x_n))\frac{\partial \hat{y}(p,\alpha,x_n)}{\partial p}\right)$$

$$\frac{\partial \hat{y}(p,\alpha,x)}{\partial p} = \sum_{h=1}^{H} \alpha_h \frac{\partial f_p(s_g(x,c_h))}{\partial p}$$

$$\frac{\partial f_p(x)}{\partial p} = \begin{cases} \frac{-(x-0.5)+a(p)-p\frac{\partial a(p)}{\partial p}}{(x-0.5-a(p))^2} - \frac{\partial a(p)}{\partial p} & \text{if } x \leq 0.5 \\ \frac{-(x-0.5)-a(p)+p\frac{\partial a(p)}{\partial p}}{(x-0.5+a(p))^2} + \frac{\partial a(p)}{\partial p} & \text{if } x \geq 0.5 \end{cases}$$

$$\frac{\partial a(p)}{\partial p} = \frac{1}{2\sqrt{2^{-4}+p}}$$

And the partial derivative of E with respect to α_i:

$$\frac{\partial E(p,\alpha)}{\partial \alpha_i} = -2\sum_{n=1}^{N}\left((y_n - \hat{y}(p,\alpha,x_n))\frac{\partial \hat{y}(p,\alpha,x_n)}{\partial \alpha_i}\right)$$

$$\frac{\partial \hat{y}(p,\alpha,x)}{\partial \alpha_i} = \begin{cases} 1 & \text{if } i = 0 \\ f_p(s_g(x,p_i)) & \text{if } i \geq 1 \end{cases}$$

A.1.2 Binomial classification

For binomial classification problems, the error function used is the cross-entropy for two classes. In this case, y_n is 1 when the observation n is positive and 0 otherwise. Functions that are the same than the regression case (e.g. f_p, $\partial f_p/\partial p$,...) are not included in this section in order to do not repeat formulas and its derivatives. So, E is defined as follows:

$$E(p,\alpha) = -\sum_{n=1}^{N}\left(y_n \cdot ln(\hat{y}(p,\alpha,x_n)) + (1-y_n)\cdot ln(1-\hat{y}(p,\alpha,x_n))\right.$$

$$\hat{y}_k(p,\alpha,x) = sigmoid\left(\xi(p,\alpha,x)\right) \qquad sigmoid(x) = \frac{1}{1+e^{-x}}$$

$$\xi(p,\alpha,x) = \sum_{h=1}^{H}\alpha_h f_p\left(s_g(x,c_h)\right) + \alpha_0$$

Derivatives

The partial derivative of E with respect to p is:

$$\frac{\partial E(p,\alpha)}{\partial p} = -\sum_{n=1}^{N}\left(y_n\frac{1}{\hat{y}(p,\alpha,x_n)}\frac{\partial \hat{y}(p,\alpha,x_n)}{\partial p} - (1-y_n)\frac{1}{1-\hat{y}(p,\alpha,x_n)}\frac{\partial \hat{y}(p,\alpha,x_n)}{\partial p}\right)$$

$$\frac{\partial \hat{y}(p,\alpha,x)}{\partial p} = \partial sigmoid\left(\xi(p,\alpha,x)\right)\frac{\partial \xi(p,\alpha,x)}{\partial p} \qquad \partial sigmoid(x) = \frac{-e^{-x}}{1+e^{-x}}$$

$$\frac{\partial \xi(p,\alpha,x)}{\partial p} = \sum_{h=1}^{H}\alpha_h\frac{\partial f_p(s_g(x,c_h))}{\partial p}$$

And the partial derivative with respect to α_i is:

$$\frac{\partial E(p,\alpha)}{\partial \alpha_i} = -\sum_{n=1}^{N}\left(y_n\frac{1}{\hat{y}(p,\alpha,x_n)}\frac{\partial \hat{y}(p,\alpha,x_n)}{\partial \alpha_i} - (1-y_n)\frac{1}{1-\hat{y}(p,\alpha,x_n)}\frac{\partial \hat{y}(p,\alpha,x_n)}{\partial \alpha_i}\right)$$

$$\frac{\partial \hat{y}(p,\alpha,x)}{\partial \alpha_i} = \partial sigmoid\left(\xi(p,\alpha,x)\right)\frac{\partial \xi(p,\alpha,x)}{\partial \alpha_i}$$

$$\frac{\partial \xi(p,\alpha,x)}{\partial \alpha_i} = \begin{cases} 1 & \text{if } i = 0 \\ f_p(s_g(x,c_i)) & \text{if } i \geq 1 \end{cases}$$

A.1.3 Multinomial classification

For multinomial classification problems, it is used the cross-entropy for k classes. Here, K is the number of modalities of the response variable, y_{nk} is 1 when the observation n is of the k modality and 0 otherwise and α is a matrix of dimensions $H+1$ times the number of modalities (K). Again, functions that were defined in previous cases are not included. The error function is defined as follows:

$$E(p,\alpha) = -\sum_{n=1}^{N}\left(\sum_{k=1}^{K} y_{nk}\cdot ln(\hat{y}_k(p,\alpha,x_n))\right)$$

$$\hat{y}_k(p,\alpha,x) = \frac{e^{\xi_k(p,\alpha,x)}}{\sum\limits_{l=1}^{K} e^{\xi_l(p,\alpha,x)}}$$

$$\xi_k(p,\alpha,x) = \begin{cases} 0 & \text{if } k=1 \\ \sum\limits_{h=1}^{H} \alpha_{hk} f_p\left(s_g(x,c_h)\right) + \alpha_{0k} & \text{if } k>1 \end{cases}$$

Derivatives

The partial derivative of E with respect to p is:

$$\frac{\partial E(p,\alpha)}{\partial p} = -\sum_{n=1}^{N}\sum_{k=1}^{K}\left(y_{nk}\frac{1}{\hat{y}_k(p,\alpha,x_n)}\frac{\partial \hat{y}_k(p,\alpha,x_n)}{\partial p}\right)$$

$$\frac{\partial \hat{y}_k(p,\alpha,x)}{\partial p} = \frac{e^{\xi_k(p,\alpha,x)}\frac{\partial \xi_k(p,\alpha,x)}{\partial p}\sum\limits_{l=1}^{K} e^{\xi_l(p,\alpha,x)} - e^{\xi_k(p,\alpha,x)}\sum\limits_{l=1}^{K}\left(e^{\xi_l(p,\alpha,x)}\frac{\partial \xi_l(p,\alpha,x)}{\partial p}\right)}{\left(\sum\limits_{l=1}^{K} e^{\xi_l(p,\alpha,x)}\right)^2}$$

$$\frac{\partial \xi_k(p,\alpha,x)}{\partial p} = \begin{cases} 0 & \text{if } k=1 \\ \sum\limits_{h=1}^{H} \alpha_{hk}\frac{\partial f_p(s_g(x,c_h))}{\partial p} & \text{if } k>1 \end{cases}$$

And the partial derivative with respect to α_{ij} is:

$$\frac{\partial E(p,\alpha)}{\partial \alpha_{ij}} = -\sum_{n=1}^{N}\sum_{k=1}^{K}\left(y_{nk}\frac{1}{\hat{y}_k(p,\alpha,x_n)}\frac{\partial \hat{y}_k(p,\alpha,x_n)}{\partial \alpha_{ij}}\right)$$

$$\frac{\partial \hat{y}_k(p,\alpha,x)}{\partial \alpha_{ij}} = \frac{e^{\xi_k(p,\alpha,x)}\frac{\partial \xi_k(p,\alpha,x)}{\partial \alpha_{ij}}\sum\limits_{l=1}^{K} e^{\xi_l(p,\alpha,x)} - e^{\xi_k(p,\alpha,x)}\sum\limits_{l=1}^{K}\left(e^{\xi_l(p,\alpha,x)}\frac{\partial \xi_l(p,\alpha,x)}{\partial \alpha_{ij}}\right)}{\left(\sum\limits_{l=1}^{K} e^{\xi_l(p,\alpha,x)}\right)^2}$$

$$\frac{\partial \xi_k(p,\alpha,x)}{\partial \alpha_{ij}} = \begin{cases} 0 & \text{if } k=1 \text{ or } k\neq j \\ 1 & \text{if } k>1 \text{ and } k=j \text{ and } i=0 \\ f_p(s_g(x,c_i)) & \text{if } k>1 \text{ and } k=j \text{ and } i>0 \end{cases}$$

A.2 MoE: Error function and its derivatives

The Ensemble of SNNs has one variant that uses the MoE method as ensemble learner. As said in section 4.2.2, this method is trained with the gradient descent algorithm, so the functions to be minimized and its derivatives should be provided. In this appendix, it is explained and formulated in details the error function (the one that should be minimized) and its derivatives. Each type of problems (regression, binomial and multinomial classification) has a different error function, so each one should be treated separately:

A.2.1 Regression

For regression problems, it is used the SSE as the error function. It is defined N as the number of observations, M as the number of SNNs used in the ensemble, P as the number of features of each observation, x_n is the vector of features for observation n (for simplicity, it is set $x_{n0} = 1$, as the intercept), y_n is the response variable for observation n and $\nu_m(x)$ is the prediction of the m SNN of the ensemble for observation x. Then, the error function (E) has one parameter Φ, a matrix of dimensions $P + 1$ times M that defines how is the gating network (thus, how to compute the β_m of the ensemble). E is defined as follows:

$$E(\Phi) = \sum_{n=1}^{N} (y_n - \hat{y}(\Phi, x_n))^2$$

$$\hat{y}(\Phi, x) = \sum_{m=1}^{M} \beta_m(\Phi, x) \cdot \nu_m(x)$$

$$\beta_m(\Phi, x) = \frac{e^{\delta_m(\Phi_{m\cdot}, x)}}{\sum_{l=1}^{M} e^{\delta_l(\Phi_{l\cdot}, x)}}$$

$$\delta_m(\Phi_{m\cdot}, x) = \sum_{p=0}^{P} \Phi_{mp} x_p$$

Derivative

The partial derivative of the previous error function with respect to Φ_{ij} (a position in the Φ matrix) is:

$$\frac{\partial E(\Phi)}{\partial \Phi_{ij}} = -2 \sum_{n=1}^{N} \left((y_n - \hat{y}(\Phi, x_n)) \frac{\partial \hat{y}(\Phi, x_n)}{\partial \Phi_{ij}} \right)$$

$$\frac{\partial \hat{y}(\Phi, x)}{\partial \Phi_{ij}} = \sum_{j=1}^{M} \frac{\partial \beta_m(\Phi, x)}{\partial \Phi_{ij}} \cdot \nu_m(x)$$

$$\frac{\partial \beta_m(\Phi, x)}{\partial \Phi_{ij}} = \begin{cases} \dfrac{e^{\delta_m(\Phi_{m\cdot}, x)} \cdot x_j \cdot \sum_{l=1}^{M} e^{\delta_l(\Phi_{l\cdot}, x)} - e^{\delta_i(\Phi_{i\cdot}, x)} \cdot e^{\delta_i(\Phi_{i\cdot}, x)} \cdot x_j}{\left(\sum_{l=1}^{M} e^{\delta_l(\Phi_{l\cdot}, x)} \right)^2} & \text{if } m = i \\ \dfrac{-e^{\delta_i(\Phi_{i\cdot}, x)} \cdot e^{\delta_i(\Phi_{i\cdot}, x)} \cdot x_j}{\left(\sum_{l=1}^{M} e^{\delta_l(\Phi_{l\cdot}, x)} \right)^2} & \text{if } m \neq i \end{cases}$$

This error function and its derivative will be used by the gradient descent algorithm to train the gating network coefficients (Φ).

A.2.2 Binomial classification

For binomial classification problems, the error function used is the cross-entropy for two classes. All constants are the same than the regression case, except y_n that is 1 when the observation n is positive and 0 otherwise and for this case, $\nu_m(x)$ is a probability. The error function is defined as follows ($\beta_m(\cdot,\cdot)$ is the same function than the regression case):

$$E(\Phi) = -\sum_{n=1}^{N} (y_n ln(\hat{y}(\Phi, x_n)) + (1 - y_n) \cdot ln(1 - \hat{y}(\Phi, x_n))$$

$$\hat{y}(\Phi, x) = \sum_{m=1}^{M} \beta_m(\Phi, x) \cdot \nu_m(x)$$

Derivative

And the partial derivative of E with respect to Φ_{ij} is:

$$\frac{\partial E(\Phi)}{\partial \Phi_{ij}} = -\sum_{n=1}^{N} \left(y_n \frac{1}{\hat{y}(\Phi, x_n)} \frac{\partial \hat{y}(\Phi, x_n)}{\partial \Phi_{ij}} + (1 - y_n) \frac{1}{1 - \hat{y}(\Phi, x_n)} \frac{-\partial \hat{y}(\Phi, x_n)}{\partial \Phi_{ij}} \right)$$

$$\frac{\partial \hat{y}(\Phi, x)}{\partial \Phi_{ij}} = \sum_{j=1}^{M} \frac{\partial \beta_m(\Phi, x)}{\partial \Phi_{ij}} \cdot \nu_m(x)$$

A.2.3 Multinomial classification

Finally, for the multinomial case, it is used the cross-entropy for k classes. All constants are the same than the regression case, except y_{nk} that is 1 when the observation n is of the k modality and $\nu_{mk}(x)$ that is the probability predicted by the m SNN of being modality k. The error function is defined as follows ($\beta_m(\cdot,\cdot)$ is the same function than the regression case):

$$E(\Phi) = -\sum_{n=1}^{N} \left(\sum_{k=1}^{K} y_{nk} \cdot ln(\hat{y}_k(\Phi, x_n)) \right)$$

$$\hat{y}_k(\Phi, x) = \sum_{m=1}^{M} \beta_m(\Phi, x) \cdot \nu_{mk}(x)$$

Derivative

The partial derivative of E with respect to Φ_{ij} is:

$$\frac{\partial E(\Phi)}{\partial \Phi_{ij}} = -\sum_{n=1}^{N} \sum_{k=1}^{K} \left(y_{nk} \frac{1}{\hat{y_k}(\Phi, x_n)} \frac{\partial \hat{y_k}(\Phi, x_n)}{\partial \Phi_{ij}} \right)$$

$$\frac{\partial \hat{y_k}(\Phi, x)}{\partial \Phi_{ij}} = \sum_{j=1}^{M} \frac{\partial \beta_m(\Phi, x)}{\partial \Phi_{ij}} \cdot \nu_{mk}(x)$$

Appendix B

Experiment results

In this appendix, it is shown the numerical results of all experiments used to test the behaviour of the SNN and the Ensemble of SNNs. See Section 6 for more details about these experiments. For each experiment, several repetitions are executed and it is shown the mean and Standard Deviation (SD) of the performance metric and the execution time of these repetitions. The following tables contain these results.

B.1 Experiment 1

Regression problems

Dataset	Learner	P method	Clustering	Reg	Mean NRMSE	Sd NRMSE	Mean time (s)	Sd time (s)
Automobile	SNN	PAM	Const	Yes	0.2197	0.0729	0.0526	0.0804
Automobile	SNN	PAM	Const	No	0.2240	0.0726	0.0381	0.0050
Automobile	SNN	Random	Const	Yes	0.1902	0.0581	0.0425	0.0415
Automobile	SNN	Random	Const	No	0.2032	0.0685	0.0280	0.0025
Automobile	SNN	PAM	Opt	No	0.1951	0.0654	0.0738	0.0256
Automobile	SNN	Random	Opt	No	0.2028	0.0869	0.0553	0.0183
Automobile	SNN	PAM	GCV	Yes	0.1937	0.0659	0.1144	0.0410
Automobile	SNN	PAM	GCV	No	0.1929	0.0669	0.0916	0.0074
Automobile	SNN	Random	GCV	Yes	0.1657	0.0559	0.0833	0.0038
Automobile	SNN	Random	GCV	No	0.1700	0.0587	0.0804	0.0029
Automobile	SNN	PAM	CV	Yes	0.1949	0.0653	0.5869	0.0371
Automobile	SNN	PAM	CV	No	0.1942	0.0658	0.5727	0.0171
Automobile	SNN	Random	CV	Yes	0.1608	0.0539	0.5696	0.0202
Automobile	SNN	Random	CV	No	0.1645	0.0524	0.5679	0.0224
Automobile	Decision tree				0.3174	0.0923	0.0071	0.0097
AutoMPG	SNN	PAM	Const	Yes	0.1577	0.0243	0.1427	0.0356
AutoMPG	SNN	PAM	Const	No	0.1600	0.0234	0.1383	0.0324
AutoMPG	SNN	Random	Const	Yes	0.1647	0.0251	0.0190	0.0016
AutoMPG	SNN	Random	Const	No	0.1692	0.0257	0.0152	0.0013
AutoMPG	SNN	PAM	Opt	No	0.1461	0.0266	0.2006	0.0431
AutoMPG	SNN	Random	Opt	No	0.1492	0.0233	0.0757	0.0197
AutoMPG	SNN	PAM	GCV	Yes	0.1388	0.0214	0.2118	0.0398
AutoMPG	SNN	PAM	GCV	No	0.1427	0.0232	0.2089	0.0412
AutoMPG	SNN	Random	GCV	Yes	0.1411	0.0197	0.0918	0.0106
AutoMPG	SNN	Random	GCV	No	0.1440	0.0202	0.0874	0.0102
AutoMPG	SNN	PAM	CV	Yes	0.1386	0.0210	0.8658	0.1108
AutoMPG	SNN	PAM	CV	No	0.1423	0.0225	0.8585	0.1090
AutoMPG	SNN	Random	CV	Yes	0.1409	0.0195	0.7479	0.0881
AutoMPG	SNN	Random	CV	No	0.1437	0.0200	0.7396	0.0904
AutoMPG	Decision tree				0.2364	0.0436	0.0030	0.0010
Communities	SNN	PAM	Const	Yes	0.3541	0.0255	45.1189	8.5145
Communities	SNN	PAM	Const	No	0.3700	0.0269	45.2132	8.6409
Communities	SNN	Random	Const	Yes	0.3545	0.0261	0.6632	0.0421
Communities	SNN	Random	Const	No	0.3713	0.0269	0.5423	0.0310
Communities	SNN	PAM	Opt	No	0.3652	0.0266	46.9140	8.7432
Communities	SNN	Random	Opt	No	0.3681	0.0266	2.4588	0.1897
Communities	SNN	PAM	GCV	Yes	0.3544	0.0258	47.5337	8.8617
Communities	SNN	PAM	GCV	No	0.3700	0.0270	47.3813	8.8057
Communities	SNN	Random	GCV	Yes	0.3543	0.0261	2.8630	0.3140
Communities	SNN	Random	GCV	No	0.3705	0.0257	2.7549	0.2847
Communities	SNN	PAM	CV	Yes	0.3542	0.0255	58.9447	9.3545
Communities	SNN	PAM	CV	No	0.3699	0.0271	58.5257	9.0593
Communities	SNN	Random	CV	Yes	0.3539	0.0263	14.4870	1.3575
Communities	SNN	Random	CV	No	0.3691	0.0260	14.3668	1.3338
Communities	Decision tree				0.6202	0.0582	0.0290	0.0206
MV	SNN	Random	Const	Yes	0.0039	0.0002	1051.2386	33.2876
MV	SNN	Random	Const	No	0.0040	0.0002	218.0586	5.5858
MV	SNN	Random	Opt	No	0.0047	0.0046	1084.1595	37.7980
MV	Decision tree				0.0755	0.0036	0.1615	0.0191

Table B.1: Results of experiment 1 for regression problems.

Binomial classification problems

Dataset	Learner	P method	Clustering	Reg	Mean accuracy	Sd accuracy	Mean time (s)	Sd time (s)
Heart	SNN	PAM	Const	Yes	82.6667	2.9865	0.2701	0.1043
Heart	SNN	PAM	Const	No	81.2444	3.2347	0.0589	0.0132
Heart	SNN	Random	Const	Yes	82.8889	3.0697	0.2266	0.0338
Heart	SNN	Random	Const	No	81.2444	3.3720	0.0291	0.0090
Heart	SNN	PAM	Opt	No	80.1778	3.8119	0.1436	0.0400
Heart	SNN	Random	Opt	No	81.3111	3.6035	0.1128	0.0281
Heart	SNN	PAM	CV	Yes	82.7556	2.6678	2.0654	0.1744
Heart	SNN	PAM	CV	No	80.8667	3.1848	1.9028	0.1976
Heart	SNN	Random	CV	Yes	82.6889	2.8930	2.1156	0.2372
Heart	SNN	Random	CV	No	81.5778	3.1674	1.8127	0.1537
Heart	Decision tree				75.2667	5.2875	0.0096	0.0083
Pima	SNN	PAM	Const	Yes	76.5625	2.4554	1.9433	0.3757
Pima	SNN	PAM	Const	No	75.1016	2.5177	1.2182	0.3294
Pima	SNN	Random	Const	Yes	76.2031	2.6267	0.8206	0.1153
Pima	SNN	Random	Const	No	74.6172	2.6487	0.0683	0.0131
Pima	SNN	PAM	Opt	No	74.9688	2.8111	1.5845	0.3673
Pima	SNN	Random	Opt	No	75.0781	2.3950	0.4435	0.0675
Pima	SNN	PAM	CV	Yes	76.8672	2.4797	8.5259	1.4588
Pima	SNN	PAM	CV	No	75.7812	2.6103	7.7789	1.4024
Pima	SNN	Random	CV	Yes	76.3438	2.3848	7.3880	1.2276
Pima	SNN	Random	CV	No	75.0938	2.4573	6.6227	1.1747
Pima	Decision tree				71.8047	2.1112	0.0103	0.0074
HorseColic2	SNN	PAM	Const	Yes	81.9180	2.8893	0.3284	0.0359
HorseColic2	SNN	PAM	Const	No	80.5246	3.1792	0.0824	0.0257
HorseColic2	SNN	Random	Const	Yes	81.7213	3.1124	0.3033	0.0338
HorseColic2	SNN	Random	Const	No	79.4098	3.4325	0.0465	0.0087
HorseColic2	SNN	PAM	Opt	No	80.6066	3.3351	0.1752	0.0447
HorseColic2	SNN	Random	Opt	No	79.5410	3.6430	0.1465	0.0349
HorseColic2	SNN	PAM	CV	Yes	82.0984	3.0819	2.5273	0.4254
HorseColic2	SNN	PAM	CV	No	81.2131	3.5278	2.2620	0.3605
HorseColic2	SNN	Random	CV	Yes	82.0328	3.3566	2.4625	0.3759
HorseColic2	SNN	Random	CV	No	80.0820	3.8276	2.2324	0.3911
HorseColic2	Decision tree				63.2131	3.6229	0.0093	0.0081
Mammographic	SNN	PAM	Const	Yes	80.7040	1.6284	2.6667	0.6637
Mammographic	SNN	PAM	Const	No	78.1869	1.9838	1.6530	0.4916
Mammographic	SNN	Random	Const	Yes	80.3925	1.7627	1.1127	0.1286
Mammographic	SNN	Random	Const	No	78.6293	2.1082	0.1211	0.0372
Mammographic	SNN	PAM	Opt	No	77.8816	2.1666	2.2101	0.5524
Mammographic	SNN	Random	Opt	No	78.3427	1.8926	0.6769	0.1002
Mammographic	SNN	PAM	CV	Yes	80.7601	1.7093	13.5135	2.6256
Mammographic	SNN	PAM	CV	No	78.1059	1.9562	12.5269	2.5388
Mammographic	SNN	Random	CV	Yes	80.6231	1.6472	11.9535	2.6059
Mammographic	SNN	Random	CV	No	78.1869	2.0688	10.9847	2.5194
Mammographic	Decision tree				78.5234	2.0925	0.0112	0.0076
Mushroom	SNN	Random	Const	Yes	96.5399	2.4007	39.0778	23.1092
Mushroom	SNN	Random	Const	No	99.4025	2.7142	18.7568	16.1348
Mushroom	SNN	Random	Opt	No	99.5074	2.6820	38.6725	29.5699
Mushroom	Decision tree				82.7518	0.5398	0.0524	0.0093
Census	SNN	Random	Const	Yes	94.7008	0.0895	7629.1551	155.5341
Census	SNN	Random	Const	No	95.1260	0.1135	2764.8296	144.8005
Census	SNN	Random	Opt	No	95.1260	0.1135	3495.6018	275.3437
Census	Decision tree				93.8663	0.0841	18.3648	2.4548

Table B.2: Results of experiment 1 for binomial classification problems.

Multinomial classification problems

Dataset	Learner	P method	Clustering	Reg	Mean accuracy	Sd accuracy	Mean time (s)	Sd time (s)
Audiology	SNN	PAM	Const	Yes	82.2368	3.9272	0.8802	0.1307
Audiology	SNN	PAM	Const	No	78.0000	5.1891	0.0630	0.0184
Audiology	SNN	Random	Const	Yes	81.2368	3.7764	0.8479	0.0583
Audiology	SNN	Random	Const	No	77.9474	4.7817	0.0503	0.0192
Audiology	SNN	PAM	Opt	No	78.7632	4.5426	0.1616	0.0471
Audiology	SNN	Random	Opt	No	78.7632	4.1097	0.1603	0.0403
Audiology	SNN	PAM	CV	Yes	82.0789	3.8604	6.0116	2.5279
Audiology	SNN	PAM	CV	No	78.8947	4.3839	5.2201	2.5328
Audiology	SNN	Random	CV	Yes	81.8684	3.8549	6.5767	3.0590
Audiology	SNN	Random	CV	No	77.4211	4.8608	5.7795	3.0758
Audiology	Decision tree				66.6316	8.9271	0.0094	0.0076
Glass	SNN	PAM	Const	Yes	52.4167	7.3964	1.0834	0.1746
Glass	SNN	PAM	Const	No	63.0833	5.6545	0.1290	0.0336
Glass	SNN	Random	Const	Yes	46.7222	9.4064	1.1142	0.1013
Glass	SNN	Random	Const	No	61.9444	7.0430	0.1406	0.0175
Glass	SNN	PAM	Opt	No	65.2500	6.0670	0.2571	0.0488
Glass	SNN	Random	Opt	No	62.5556	7.1040	0.2586	0.0357
Glass	SNN	PAM	CV	Yes	53.7500	7.8379	24.5722	1.3746
Glass	SNN	PAM	CV	No	63.0278	7.4286	23.4915	1.4401
Glass	SNN	Random	CV	Yes	48.9722	8.5746	24.7046	0.7706
Glass	SNN	Random	CV	No	61.6944	7.0602	23.7568	0.7481
Glass	Decision tree				64.5278	4.7305	0.0074	0.0015
HorseColic1	SNN	PAM	Const	Yes	66.0165	4.2425	0.9590	0.1058
HorseColic1	SNN	PAM	Const	No	63.5207	4.1340	0.0727	0.0187
HorseColic1	SNN	Random	Const	Yes	66.6612	4.3238	0.9396	0.0923
HorseColic1	SNN	Random	Const	No	64.2314	4.1374	0.0587	0.0080
HorseColic1	SNN	PAM	Opt	No	64.1488	3.8580	0.2320	0.0469
HorseColic1	SNN	Random	Opt	No	64.3802	4.2050	0.1927	0.0328
HorseColic1	SNN	PAM	CV	Yes	66.0661	4.0742	6.9060	1.5885
HorseColic1	SNN	PAM	CV	No	64.6446	4.4436	6.0123	1.4630
HorseColic1	SNN	Random	CV	Yes	66.7934	4.2787	7.2541	2.2014
HorseColic1	SNN	Random	CV	No	64.6446	3.7717	6.3374	2.1201
HorseColic1	Decision tree				47.1570	14.2230	0.0097	0.0029
Annealing	SNN	PAM	Const	Yes	92.2216	1.8833	6.9006	1.1292
Annealing	SNN	PAM	Const	No	93.2891	1.8012	1.7261	0.4763
Annealing	SNN	Random	Const	Yes	90.8618	1.9075	5.4127	0.6933
Annealing	SNN	Random	Const	No	92.7132	2.4282	0.2256	0.0461
Annealing	SNN	PAM	Opt	No	96.8697	1.5182	2.0741	0.5095
Annealing	SNN	Random	Opt	No	96.4784	1.4268	0.5796	0.0888
Annealing	SNN	PAM	CV	Yes	92.2843	1.9441	31.4569	3.8603
Annealing	SNN	PAM	CV	No	93.3819	1.8298	26.3559	3.4946
Annealing	SNN	Random	CV	Yes	91.1124	1.9620	31.6393	5.6533
Annealing	SNN	Random	CV	No	93.1611	1.8873	26.5291	5.2667
Annealing	Decision tree				76.4887	1.8685	0.0106	0.0052
Contraceptive	SNN	PAM	Const	Yes	52.1100	2.0354	40.7249	8.7936
Contraceptive	SNN	PAM	Const	No	50.5662	2.0040	30.6197	7.5759
Contraceptive	SNN	Random	Const	Yes	53.2261	1.8596	8.0556	0.8083
Contraceptive	SNN	Random	Const	No	51.2912	1.8064	0.9070	0.2442
Contraceptive	SNN	PAM	Opt	No	51.7271	2.2399	32.0491	8.2886
Contraceptive	SNN	Random	Opt	No	51.9430	2.0312	2.4014	0.4170
Contraceptive	SNN	PAM	CV	Yes	52.5784	2.2255	229.6423	44.0599
Contraceptive	SNN	PAM	CV	No	50.9043	1.9897	221.7207	44.1901
Contraceptive	SNN	Random	CV	Yes	54.2525	2.1889	202.8838	38.5094
Contraceptive	SNN	Random	CV	No	51.6334	2.0917	196.9093	37.9923
Contraceptive	Decision tree				52.1263	2.4867	0.0171	0.0036
Diabetis	SNN	Random	Const	Yes	58.8792	0.1476	5216.2397	198.5430
Diabetis	SNN	Random	Const	No	58.7518	0.2443	3371.3839	522.3402
Diabetis	SNN	Random	Opt	No	58.7518	0.2443	3574.4267	323.4258
Diabetis	Decision tree				55.9796	0.1556	13442.1101	477.1944

Table B.3: Results of experiment 1 for multinomial classification problems.

B.2 Experiment 2

Regression problems

Dataset	Learner	Ensemble method	Mean NRMSE	Sd NRMSE	Mean time (s)	Sd time (s)
Automobile	Ens SNN	A	0.1349	0.0372	1.7625	0.1806
Automobile	Ens SNN	B	0.9056	0.6789	2.2777	0.0838
Automobile	Ens SNN	B2	0.1556	0.0948	2.2899	0.0716
Automobile	Ens SNN	C	0.2189	0.1077	26.6964	5.6131
Automobile	Ens SNN	C2	0.9494	1.3950	5.0988	2.4856
Automobile	Random Forest		0.0843	0.0455	0.1613	0.0125
AutoMPG	Ens SNN	A	0.1488	0.0232	0.9330	0.0763
AutoMPG	Ens SNN	B	0.2292	0.0704	1.2813	0.0790
AutoMPG	Ens SNN	B2	0.1470	0.0256	1.2893	0.0681
AutoMPG	Ens SNN	C	0.1626	0.0290	15.7269	3.1808
AutoMPG	Ens SNN	C2	0.6552	1.6560	4.3804	1.5239
AutoMPG	Random Forest		0.1344	0.0229	0.1377	0.0331
Communities	Ens SNN	A	0.3530	0.0252	8.5382	0.3597
Communities	Ens SNN	B	0.3691	0.0301	14.1528	0.4414
Communities	Ens SNN	B2	0.3510	0.0262	14.3569	0.4400
Communities	Ens SNN	C	0.3691	0.0281	139.7070	30.6660
Communities	Ens SNN	C2	0.4340	0.1974	86.4497	16.3308
Communities	Random Forest		0.3540	0.0262	13.7014	0.1083
MV	Ens SNN	A	0.0309	0.0017	285.1223	7.3354
MV	Ens SNN	B	0.0097	0.0002	691.6962	17.1160
MV	Ens SNN	B2	0.0097	0.0002	672.2314	17.6960
MV	Random Forest		0.0011	0.0001	686.1082	11.9656

Table B.4: Results of experiment 2 for regression problems.

Binomial classification problems

Dataset	Learner	Ensemble method	Mean accuracy	Sd accuracy	Mean time (s)	Sd time (s)
Heart	Ens SNN	A	83.1111	2.7309	2.2540	0.3972
Heart	Ens SNN	A2	83.1111	2.7031	2.1069	0.1474
Heart	Ens SNN	B	72.4667	5.5772	2.7992	0.2496
Heart	Ens SNN	B2	82.1333	3.0519	3.2319	0.2567
Heart	Ens SNN	C	76.6889	4.2562	9.1640	2.4805
Heart	Ens SNN	C2	80.7556	3.5524	3.0980	0.2528
Heart	Random Forest		81.6222	3.2078	0.0926	0.0098
Pima	Ens SNN	A	76.4266	2.3872	2.2878	0.1444
Pima	Ens SNN	A2	76.1094	2.2891	2.2740	0.1894
Pima	Ens SNN	B	72.3516	2.6556	3.3634	0.2842
Pima	Ens SNN	B2	76.3281	2.3649	4.4769	0.2498
Pima	Ens SNN	C	74.7578	2.4905	27.7608	11.0826
Pima	Ens SNN	C2	72.7031	2.4127	14.4129	0.9594
Pima	Random Forest		76.1328	2.4289	0.2550	0.0164
HorseColic2	Ens SNN	A	81.8689	2.8665	2.7700	0.1798
HorseColic2	Ens SNN	A2	81.9344	3.1766	2.7815	0.2043
HorseColic2	Ens SNN	B	70.8361	3.9885	3.8689	0.2397
HorseColic2	Ens SNN	B2	81.4590	3.4968	4.4130	0.3093
HorseColic2	Ens SNN	C	77.0328	3.9761	12.2150	2.9719
HorseColic2	Ens SNN	C2	70.6230	6.6577	3.6905	0.5558
HorseColic2	Random Forest		84.4590	3.3774	0.1855	0.0151
Mammographic	Ens SNN	A	80.4174	1.6923	2.8633	0.1893
Mammographic	Ens SNN	A2	80.3614	1.7543	2.9024	0.2628
Mammographic	Ens SNN	B	75.9003	2.2890	4.3730	0.2681
Mammographic	Ens SNN	B2	80.2243	1.7648	5.6558	0.2961
Mammographic	Ens SNN	C	78.2741	1.8152	32.1568	7.9243
Mammographic	Ens SNN	C2	76.3676	2.6102	17.4076	1.7076
Mammographic	Random Forest		78.7477	1.6833	0.2205	0.0167
Mushroom	Ens SNN	A	99.9834	0.0305	196.9214	15.4798
Mushroom	Ens SNN	A2	99.9852	0.0326	196.9838	15.1680
Mushroom	Ens SNN	B	99.9760	0.0345	253.0309	18.3923
Mushroom	Ens SNN	B2	99.9889	0.0296	258.9538	18.3639
Mushroom	Ens SNN	C	99.9317	0.0712	1622.5809	644.7956
Mushroom	Ens SNN	C2	70.1440	9.5435	1073.4488	149.9142
Mushroom	Random Forest		100.0000	0.0000	1.0169	0.1116
Census	Ens SNN	A	94.6773	0.0518	1814.0934	132.9791
Census	Ens SNN	A2	94.7675	0.0521	1865.4913	125.4543
Census	Ens SNN	B	94.9639	0.0802	4835.5152	168.3627
Census	Ens SNN	B2	93.7433	0.0331	5043.4455	222.4711
Census	Random Forest		95.5190	0.0503	296.0602	2.7437

Table B.5: Results of experiment 2 for binomial classification problems.

Multinomial classification problems

Dataset	Learner	Ensemble method	Mean accuracy	Sd accuracy	Mean time (s)	Sd time (s)
Audiology	Ens SNN	A	85.6842	3.3270	3.1249	0.4251
Audiology	Ens SNN	A2	85.7632	3.4218	2.9168	0.1953
Audiology	Ens SNN	B	85.7105	3.3940	4.2393	0.3332
Audiology	Ens SNN	B2	83.7632	3.3161	9.7934	0.4102
Audiology	Ens SNN	C	78.1579	5.0856	8.8731	1.2903
Audiology	Ens SNN	C2	80.8421	4.7416	3.6157	0.2667
Audiology	Random Forest	RandForest	85.2368	3.6185	0.1108	0.0293
Glass	Ens SNN	A	70.9722	5.3773	3.7178	0.5424
Glass	Ens SNN	A2	70.9722	5.5431	3.6203	0.4844
Glass	Ens SNN	B	68.9444	5.8709	14.7871	3.7414
Glass	Ens SNN	B2	71.2500	6.0198	14.0818	0.7083
Glass	Ens SNN	C	64.9167	6.6450	11.3334	2.4195
Glass	Ens SNN	C2	62.5556	6.0113	2.1334	0.2699
Glass	Random Forest	RandForest	75.4167	4.8540	0.0749	0.0112
HorseColic1	Ens SNN	A	67.5041	3.9843	3.3228	0.2412
HorseColic1	Ens SNN	A2	67.5537	3.9364	3.1934	0.1973
HorseColic1	Ens SNN	B	59.4711	4.4143	4.7216	0.5143
HorseColic1	Ens SNN	B2	67.2066	4.3361	8.4611	0.4362
HorseColic1	Ens SNN	C	61.4711	4.2891	12.7660	1.6017
HorseColic1	Ens SNN	C2	60.4132	3.9872	4.1470	0.3418
HorseColic1	Random Forest	RandForest	69.7521	4.0246	0.2106	0.0135
Annealing	Ens SNN	A	95.7218	1.2385	8.7573	0.7348
Annealing	Ens SNN	A2	95.7895	1.2686	8.4854	0.6167
Annealing	Ens SNN	B	94.4361	2.2441	13.8708	1.3692
Annealing	Ens SNN	B2	92.0000	1.4845	38.8668	1.2802
Annealing	Ens SNN	C	92.7744	2.0707	26.9700	5.0711
Annealing	Ens SNN	C2	93.2180	2.5700	15.7100	1.1091
Annealing	Random Forest	RandForest	87.5038	2.1503	0.3509	0.0230
Contraceptive	Ens SNN	A	54.4969	2.0483	10.2479	0.5703
Contraceptive	Ens SNN	A2	54.6680	2.1847	9.6668	0.4466
Contraceptive	Ens SNN	B	48.1996	2.1568	15.0437	0.7461
Contraceptive	Ens SNN	B2	54.9980	2.1109	28.9676	0.8235
Contraceptive	Ens SNN	C	51.1120	2.0594	85.4389	14.9056
Contraceptive	Ens SNN	C2	49.2383	2.4837	77.4590	5.0630
Contraceptive	Random Forest	RandForest	53.2994	1.6881	0.5989	0.0607
Diabetes	Ens SNN	A	58.5113	0.2363	1997.2821	315.8804
Diabetes	Ens SNN	A2	58.6770	0.1753	2147.2768	270.9392
Diabetes	Ens SNN	B	58.8816	0.1227	4621.6094	509.7326
Diabetes	Ens SNN	B2	58.9782	0.0756	9522.7875	2133.4364
Diabetes	Random Forest	RandForest	58.9609	0.1636	2423.7258	20.2017

Table B.6: Results of experiment 2 for multinomial classification problems.

B.3 Experiment 3

For this experiment, when the percentage of observations is 0% it is used 2 prototypes (minimum case).

Regression problems

Dataset	Percentage of observations that are prototypes	Mean NRMSE	Sd NRMSE	Mean time (s)	Sd time (s)
Automobile	0%	0.4604	0.0621	0.1196	0.2393
Automobile	5%	0.3617	0.0882	0.0539	0.0131
Automobile	10%	0.1787	0.0386	0.1789	0.2549
Automobile	20%	0.1716	0.0362	0.0731	0.0330
Automobile	30%	0.1289	0.0273	0.1170	0.0206
Automobile	40%	0.1414	0.0506	0.1346	0.0131
Automobile	50%	0.1721	0.1339	0.1733	0.0507
Automobile	60%	0.2214	0.1102	0.1422	0.0615
Automobile	70%	0.8937	1.7982	0.1146	0.0507
Automobile	80%	1.0683	0.8013	0.1538	0.0657
Automobile	90%	3.4899	3.1238	0.2062	0.1184
Automobile	100%	43777.5989	137035.3460	0.1702	0.0759
AutoMPG	0%	0.2481	0.0377	0.0453	0.0068
AutoMPG	5%	0.1476	0.0174	0.1312	0.0191
AutoMPG	10%	0.1490	0.0270	0.0848	0.0386
AutoMPG	20%	0.1465	0.0316	0.2058	0.0592
AutoMPG	30%	0.1799	0.0331	0.1720	0.0959
AutoMPG	40%	0.2094	0.0523	0.2483	0.1632
AutoMPG	50%	0.3209	0.1428	0.3481	0.1966
AutoMPG	60%	0.3615	0.1590	0.4600	0.1970
AutoMPG	70%	0.3830	0.0898	0.7552	0.0994
AutoMPG	80%	0.6100	0.1677	0.8788	0.0992
AutoMPG	90%	1.3649	0.6092	0.9537	0.0757
AutoMPG	100%	2144.8579	6563.6163	1.1050	0.0702
Communities	0%	0.6130	0.0289	1.3474	0.1286
Communities	5%	0.3517	0.0269	5.2628	0.2992
Communities	10%	0.3613	0.0243	3.4063	0.1541
Communities	20%	0.3961	0.0246	7.2449	0.4356
Communities	30%	0.4538	0.0261	13.5282	4.6165
Communities	40%	0.5276	0.0401	17.8584	2.3382
Communities	50%	0.6597	0.0596	24.0280	3.1806
Communities	60%	0.8331	0.0862	28.7702	1.9799
Communities	70%	1.1290	0.1002	28.5420	0.8957
Communities	80%	1.7666	0.1896	32.0069	1.9117
Communities	90%	3.6208	0.6209	37.3002	1.5406
Communities	100%	1896.2534	2674.8342	40.4964	1.2198

Table B.7: Results of experiment 3 for regression problems.

Binomial classification problems

Dataset	Percentage of observations that are prototypes	Mean Accuracy	Sd Accuracy	Mean time (s)	Sd time (s)
Heart	0%	80.7778	2.2861	0.1396	0.2721
Heart	5%	80.4444	3.5985	0.0833	0.0192
Heart	10%	80.4444	5.1373	0.1309	0.0196
Heart	20%	77.2222	4.6036	0.2224	0.0180
Heart	30%	72.1111	7.1521	0.3095	0.1011
Heart	40%	69.1111	4.2809	0.3458	0.0525
Heart	50%	67.2222	5.6716	0.4112	0.0737
Heart	60%	67.8889	6.1853	0.4226	0.0698
Heart	70%	65.7778	3.3457	0.4648	0.1076
Heart	80%	62.8889	5.9074	0.5672	0.1133
Heart	90%	60.5556	4.8644	0.5873	0.0976
Heart	100%	51.5556	6.5672	0.6795	0.1138
Pima	0%	66.9141	3.8672	0.1862	0.0301
Pima	5%	76.0547	1.9492	0.6828	0.1735
Pima	10%	75.3125	2.6608	1.5639	0.3314
Pima	20%	72.6172	2.0293	4.2847	0.6807
Pima	30%	69.5312	2.4637	6.2652	1.1149
Pima	40%	65.7422	2.1320	9.9388	3.1984
Pima	50%	63.4375	2.0356	8.7838	1.2538
Pima	60%	62.9688	1.8761	7.8721	2.2191
Pima	70%	64.1016	4.3360	8.4740	1.3334
Pima	80%	63.4766	4.1880	7.4571	0.9407
Pima	90%	59.4922	4.8756	8.3398	0.8380
Pima	100%	55.3125	5.4195	9.7056	0.6356
HorseColic2	0%	81.5574	2.7117	0.0741	0.0256
HorseColic2	5%	81.1475	2.5631	0.1646	0.0826
HorseColic2	10%	81.3115	2.5863	0.1952	0.0610
HorseColic2	20%	75.2459	2.3756	0.2845	0.0743
HorseColic2	30%	71.0656	3.0191	0.4205	0.1283
HorseColic2	40%	70.7377	3.6667	0.7581	0.1277
HorseColic2	50%	68.9344	3.8533	0.8144	0.1041
HorseColic2	60%	67.1311	3.4872	0.9132	0.0738
HorseColic2	70%	64.5082	4.1804	1.1416	0.1041
HorseColic2	80%	60.0000	5.2099	1.7533	0.5467
HorseColic2	90%	59.2623	3.0924	1.3483	0.3338
HorseColic2	100%	56.1475	2.2941	1.3879	0.1575
Mammographic	0%	67.4143	2.2426	0.1916	0.0286
Mammographic	5%	79.0343	2.6921	1.0888	0.3588
Mammographic	10%	78.0685	1.9931	2.8387	0.5162
Mammographic	20%	72.5545	4.3625	6.6377	2.8704
Mammographic	30%	69.1589	2.2703	7.1500	1.1104
Mammographic	40%	67.7882	3.2281	6.9876	1.2195
Mammographic	50%	56.0436	16.8509	8.7539	2.8296
Mammographic	60%	59.5016	6.4349	9.9113	2.7339
Mammographic	70%	50.5607	9.7811	10.0738	3.8023
Mammographic	80%	56.4174	10.6157	14.7311	5.6882
Mammographic	90%	53.5514	13.0993	22.9377	11.2873
Mammographic	100%	57.2274	6.5347	19.3434	6.0560

Table B.8: Results of experiment 3 for binomial classification problems.

Multinomial classification problems

Dataset	Percentage of observations that are prototypes	Mean Accuracy	Sd Accuracy	Mean time (s)	Sd time (s)
Audiology	0%	71.9737	3.9741	0.0983	0.0323
Audiology	5%	82.1053	3.6800	0.1439	0.0206
Audiology	10%	79.7368	4.6500	0.2078	0.1098
Audiology	20%	78.6842	3.3860	0.2204	0.0393
Audiology	30%	78.4211	3.8336	0.2837	0.0274
Audiology	40%	78.0263	7.0463	0.4175	0.1081
Audiology	50%	79.0789	4.9796	0.3902	0.0651
Audiology	60%	80.0000	4.5496	0.4422	0.1012
Audiology	70%	79.8684	4.1632	0.5500	0.1275
Audiology	80%	80.9211	4.2183	0.5222	0.1200
Audiology	90%	78.6842	4.5071	0.5916	0.0958
Audiology	100%	79.8684	2.6352	0.7048	0.3249
Glass	0%	35.9722	5.8359	0.1281	0.0140
Glass	5%	58.7500	7.8853	0.3355	0.0972
Glass	10%	62.9167	6.7424	0.3174	0.0996
Glass	20%	63.3333	7.6129	0.3171	0.0603
Glass	30%	62.7778	4.5266	0.3738	0.1064
Glass	40%	64.8611	5.4010	0.3779	0.0185
Glass	50%	64.3056	4.4430	0.4171	0.0521
Glass	60%	65.4167	5.0056	0.5011	0.1044
Glass	70%	65.4167	6.1923	0.4988	0.0690
Glass	80%	65.5556	4.3331	0.5869	0.0589
Glass	90%	66.6667	4.3921	0.7312	0.0884
Glass	100%	68.3333	3.3894	0.7873	0.0989
HorseColic1	0%	17.0248	3.0972	0.0767	0.0174
HorseColic1	5%	64.0496	3.4682	0.1431	0.0244
HorseColic1	10%	64.1322	4.8221	0.2457	0.0641
HorseColic1	20%	61.3223	3.7124	0.3624	0.0307
HorseColic1	30%	58.7603	3.4726	0.4330	0.0403
HorseColic1	40%	58.6777	2.9670	0.4943	0.1070
HorseColic1	50%	60.9091	4.0870	0.5251	0.0594
HorseColic1	60%	60.4959	4.7524	0.6132	0.0749
HorseColic1	70%	59.8347	5.1567	0.6943	0.0674
HorseColic1	80%	58.4298	4.8668	0.6680	0.2418
HorseColic1	90%	57.6033	4.8355	0.6542	0.3732
HorseColic1	100%	56.2810	4.5509	0.5360	0.2089
Annealing	0%	2.9699	4.2586	0.2263	0.0754
Annealing	5%	95.0752	1.8959	0.9534	0.1672
Annealing	10%	97.0301	1.1410	2.0672	0.2298
Annealing	20%	97.1805	1.8692	4.4653	0.3175
Annealing	30%	88.4586	31.0878	7.7339	1.4005
Annealing	40%	88.4211	31.0852	11.8730	1.7602
Annealing	50%	97.7444	1.4614	11.9135	3.1722
Annealing	60%	87.9699	30.2937	9.1422	0.9465
Annealing	70%	88.2707	31.0255	9.0418	1.5286
Annealing	80%	88.1955	30.7440	10.7955	2.3210
Annealing	90%	96.8045	1.6742	11.8536	2.8015
Annealing	100%	96.3534	1.6817	13.3451	4.9609
Contraceptive	0%	46.0692	2.1550	0.4415	0.1093
Contraceptive	5%	52.7088	1.7750	10.0108	1.9786
Contraceptive	10%	51.4868	2.2777	29.2299	2.9630
Contraceptive	20%	47.5560	2.9157	69.3159	12.9493
Contraceptive	30%	45.1120	2.0664	105.0595	14.6734
Contraceptive	40%	44.7251	2.4044	125.9849	13.8911
Contraceptive	50%	44.6436	1.2838	118.8698	9.0113
Contraceptive	60%	44.4603	1.4657	89.3849	17.0330
Contraceptive	70%	44.6640	1.4906	65.6498	10.0515
Contraceptive	80%	44.2363	1.5083	38.4932	5.9946
Contraceptive	90%	44.8065	1.4176	25.1099	4.2761
Contraceptive	100%	44.2159	1.5255	28.5400	4.9655

Table B.9: Results of experiment 3 for multinomial classification problems.

B.4 Experiment 4

Regression problems

Dataset	Clustering method	Mean NRMSE	Sd NRMSE	Mean time (s)	Sd time (s)
Automobile	PAM	0.2135	0.0228	2.4805	1.8446
Automobile	Random	0.1989	0.0259	1.9204	0.1340
AutoMPG	PAM	0.1521	0.0229	2.5138	0.1722
AutoMPG	Random	0.1492	0.0228	0.9981	0.1828
Communities	PAM	0.3385	0.0215	348.3088	16.4894
Communities	Random	0.3374	0.0221	17.5576	1.6195

Table B.10: Results of experiment 4 for regression problems.

Classification problems

Dataset	Clustering method	Mean Accuracy	Sd Accuracy	Mean time (s)	Sd time (s)
Heart	PAM	82.3333	2.6938	2.8297	1.0472
Heart	Random	82.7778	3.1098	1.5107	0.0842
Pima	PAM	76.7188	2.0604	16.6981	3.2930
Pima	Random	76.4062	1.3927	3.9121	1.3744
HorseColic2 (bin)	PAM	82.3770	2.7930	3.4165	0.2944
HorseColic2 (bin)	Random	82.6230	3.1816	2.8254	0.2383
Mammographic	PAM	80.2181	2.0042	20.0315	2.9268
Mammographic	Random	80.4984	2.3689	3.6970	0.2289
Audiology	PAM	83.4211	2.2535	3.2946	0.2068
Audiology	Random	83.9474	2.4655	3.2342	0.1282
Glass	PAM	66.6667	8.4863	3.9227	0.2507
Glass	Random	66.2500	7.6087	4.3502	0.2082
HorseColic1	PAM	66.6116	3.9192	3.1816	0.2055
HorseColic1	Random	68.1818	4.2899	2.8375	0.1090
Annealing	PAM	96.7669	1.5672	19.1816	1.3579
Annealing	Random	96.3534	1.0492	8.4673	0.5225
Contraceptive	PAM	52.8513	2.6210	186.1039	10.6903
Contraceptive	Random	54.3788	2.0208	18.4383	0.6604

Table B.11: Results of experiment 4 for classification problems.

B.5 Experiment 5

Regression problems

Dataset	Regularization of each of the SNNs of the ensemble of SNNs	Mean NRMSE	Sd NRMSE	Mean time (s)	Sd time (s)
Automobile	No	0.1338	0.0309	2.5242	0.7415
Automobile	Yes	0.1451	0.0268	2.5350	0.4713
AutoMPG	No	0.1407	0.0187	2.1274	1.1294
AutoMPG	Yes	0.1443	0.0203	2.7399	0.5619
Communities	No	0.3456	0.0214	29.0444	4.0871
Communities	Yes	0.3518	0.0213	31.9718	5.0038

Table B.12: Results of experiment 5 for regression problems.

Classification problems

Dataset	Regularization of each of the SNNs of the ensemble of SNNs	Mean Accuracy	Sd Accuracy	Mean time (s)	Sd time (s)
Heart	No	83.9444	4.5238	2.0446	0.3873
Heart	Yes	82.8889	2.8784	17.8466	2.0061
Pima	No	76.0156	1.6388	2.4466	0.1370
Pima	Yes	76.7578	1.8529	30.6836	4.0792
HorseColic2	No	82.5410	3.7072	4.1394	1.1030
HorseColic2	Yes	83.0328	3.0924	24.3117	4.0951
Mammographic	No	80.9657	2.0998	5.4104	1.9116
Mammographic	Yes	80.4673	1.9319	42.4763	12.2666
Audiology	No	85.9211	4.6437	3.0918	0.4006
Audiology	Yes	83.8158	5.8203	60.3073	1.9731
Glass	No	68.1944	6.9737	3.7482	0.4644
Glass	Yes	61.2500	5.4169	84.4419	1.6462
HorseColic1	No	69.0083	2.1779	3.2335	0.0706
HorseColic1	Yes	66.4463	3.0228	55.9954	0.5599
Annealing	No	95.4135	0.6340	8.6029	0.3631
Annealing	Yes	84.4737	1.7003	155.9360	1.5385
Contraceptive	No	54.2159	2.7773	9.8912	0.2871
Contraceptive	Yes	51.4460	2.7429	174.9014	2.1916

Table B.13: Results of experiment 5 for classification problems.

B.6 Experiment 6

Regression problems

Dataset	Method to select the number of prototypes	Mean NRMSE	Sd NRMSE	Mean time (s)	Sd time (s)
Automobile	Uniform	0.1762	0.0325	5.9687	0.8032
Automobile	Binomial	0.1777	0.0300	6.0321	0.3876
Automobile	Poisson	0.1801	0.0298	5.9360	0.4476
Automobile	Constant	0.1859	0.0288	6.0264	0.4392
AutoMPG	Uniform	0.1463	0.0226	7.8754	1.0121
AutoMPG	Binomial	0.1421	0.0210	6.9687	0.5857
AutoMPG	Poisson	0.1431	0.0205	6.5646	0.4314
AutoMPG	Constant	0.1405	0.0204	8.2123	0.7240
Communities	Uniform	0.3388	0.0221	105.4056	5.5891
Communities	Binomial	0.3380	0.0230	105.0609	5.4359
Communities	Poisson	0.3380	0.0234	106.3258	5.0205
Communities	Constant	0.3380	0.0230	102.2761	3.4644

Table B.14: Results of experiment 6 for regression problems.

Classification problems

Dataset	Method to select the number of prototypes	Mean Accuracy	Sd Accuracy	Mean time (s)	Sd time (s)
Heart	Uniform	83.2222	3.0779	10.1522	1.0911
Heart	Binomial	83.4444	2.6953	10.2513	1.3761
Heart	Poisson	83.3889	3.2539	10.1081	0.6547
Heart	Constant	82.9444	3.1689	10.0619	0.5398
Pima	Uniform	76.3477	1.7810	29.0361	1.2641
Pima	Binomial	76.6016	1.9630	29.3074	1.5159
Pima	Poisson	76.4844	1.9902	28.8117	1.0735
Pima	Constant	76.5625	1.9304	28.5243	1.1140
HorseColic2	Uniform	82.5410	3.1923	13.9488	0.8820
HorseColic2	Binomial	82.1721	2.9599	13.6140	0.7480
HorseColic2	Poisson	82.0902	3.1943	13.6632	0.8672
HorseColic2	Constant	82.1311	2.8219	13.9477	0.6250
Mammographic	Uniform	80.7944	1.9317	37.5156	1.9585
Mammographic	Binomial	80.4673	2.1300	39.3071	1.2304
Mammographic	Poisson	80.3115	2.0728	40.7590	4.2869
Mammographic	Constant	80.1869	2.0846	39.4839	1.4873
Audiology	Uniform	83.4211	3.7557	20.7188	1.8828
Audiology	Binomial	83.7500	2.8725	21.2536	0.9989
Audiology	Poisson	83.8816	3.0747	21.2374	0.7029
Audiology	Constant	83.8158	2.8349	21.9456	0.9317
Glass	Uniform	68.6806	6.2027	25.6639	1.3802
Glass	Binomial	68.5417	6.3130	25.4454	0.8334
Glass	Poisson	68.5417	6.7182	25.0821	1.0129
Glass	Constant	68.6111	5.6173	25.9626	1.1852
HorseColic1	Uniform	66.6116	3.3747	22.6849	0.7380
HorseColic1	Binomial	66.6116	3.8615	23.1085	1.2019
HorseColic1	Poisson	66.9008	3.8248	22.6894	0.7909
HorseColic1	Constant	66.9008	3.6121	22.5357	0.7819
Annealing	Uniform	98.0075	0.8100	48.0669	1.8484
Annealing	Binomial	98.4211	0.7867	48.1127	2.1093
Annealing	Poisson	98.2519	0.8475	48.6608	1.7441
Annealing	Constant	98.1767	0.8903	47.8419	2.3039
Contraceptive	Uniform	54.7862	2.0420	119.1086	4.6548
Contraceptive	Binomial	54.3890	1.9473	113.5524	5.5602
Contraceptive	Poisson	54.8574	1.6602	111.7606	3.4926
Contraceptive	Constant	54.5927	1.6084	107.6989	2.6657

Table B.15: Results of experiment 6 for classification problems.

B.7 Experiment 7

Regression problems

Dataset	Method to choose the number of observations of each SNN (EnsSNN)	Mean NRMSE	Sd NRMSE	Mean time (s)	Sd time (s)
Automobile	Uniform	0.1321	0.0314	3.6698	0.3966
Automobile	Binomial	0.1352	0.0364	3.4226	0.2698
Automobile	Poisson	0.1311	0.0354	3.5984	0.2873
Automobile	Constant	0.1329	0.0359	3.5822	0.2688
AutoMPG	Uniform	0.1321	0.0164	4.3458	0.4705
AutoMPG	Binomial	0.1307	0.0183	4.0921	0.4275
AutoMPG	Poisson	0.1307	0.0177	4.2284	0.3945
AutoMPG	Constant	0.1316	0.0177	4.0374	0.3619
Communities	Uniform	0.3444	0.0212	54.5404	8.2804
Communities	Binomial	0.3436	0.0211	38.7829	3.5136
Communities	Poisson	0.3428	0.0210	56.4734	29.3420
Communities	Constant	0.3426	0.0211	71.5511	27.1689

Table B.16: Results of experiment 7 for regression problems.

Classification problems

Dataset	Method to choose the number of observations of each SNN (EnsSNN)	Mean Accuracy	Sd Accuracy	Mean time (s)	Sd time (s)
Heart	Uniform	82.7778	3.3230	10.2818	4.8812
Heart	Binomial	82.8889	3.5602	6.9539	1.2732
Heart	Poisson	82.2222	2.6708	6.4709	0.2134
Heart	Constant	83.1111	2.6084	6.6952	0.8033
Pima	Uniform	76.6016	1.6922	14.3119	0.8732
Pima	Binomial	76.1719	1.8136	13.1276	1.0758
Pima	Poisson	76.7578	1.9422	12.9235	1.4798
Pima	Constant	76.4453	1.9922	12.6509	0.3091
HorseColic2	Uniform	82.7869	2.1514	8.2387	0.3015
HorseColic2	Binomial	82.3770	2.8721	8.2150	0.5080
HorseColic2	Poisson	81.9672	2.9172	7.9285	0.5919
HorseColic2	Constant	82.7049	3.7750	7.7479	0.3858
Mammographic	Uniform	80.5296	2.2380	17.3152	1.4121
Mammographic	Binomial	80.4984	2.0831	15.6274	0.6198
Mammographic	Poisson	80.6542	2.1049	15.4166	0.5992
Mammographic	Constant	80.4673	1.8462	15.0540	0.7863
Audiology	Uniform	86.5789	2.6894	13.6558	2.3409
Audiology	Binomial	86.5789	2.6169	17.1046	6.1016
Audiology	Poisson	86.1842	2.9908	11.6459	0.7632
Audiology	Constant	86.3158	3.2349	11.2433	0.5378
Glass	Uniform	70.1389	7.0895	15.9809	0.9820
Glass	Binomial	68.3333	8.0932	13.9482	1.0324
Glass	Poisson	68.7500	7.2981	13.4049	0.6319
Glass	Constant	69.3056	7.3333	13.4435	0.5395
HorseColic1	Uniform	68.0992	4.2890	12.2369	0.7830
HorseColic1	Binomial	66.9421	5.1391	14.0884	1.9426
HorseColic1	Poisson	67.2727	4.0337	14.6278	3.2769
HorseColic1	Constant	66.7769	5.0316	13.6783	0.7823
Annealing	Uniform	98.2331	0.9386	33.8117	7.5612
Annealing	Binomial	98.1955	0.7886	34.2750	12.2408
Annealing	Poisson	98.2331	0.7935	28.7016	2.3282
Annealing	Constant	97.8571	0.9552	28.3021	4.1688
Contraceptive	Uniform	54.9084	1.5337	63.2700	9.4626
Contraceptive	Binomial	54.3381	1.3672	53.0528	7.9359
Contraceptive	Poisson	54.7454	1.3706	46.7019	4.7160
Contraceptive	Constant	54.8269	1.4395	40.3186	2.4972

Table B.17: Results of experiment 7 for classification problems.

B.8 Experiment 8

Regression problems

Dataset	Proportion of observations used to train each SNN (EnsSNN)	Mean NRMSE	Sd NRMSE	Mean time (s)	Sd time (s)
Automobile	0%	288.8448	871.4941	1.9334	0.1012
Automobile	10%	28.2060	62.4932	1.9394	0.1574
Automobile	20%	0.1490	0.0330	2.1526	0.1414
Automobile	30%	0.1444	0.0311	2.3232	0.5456
Automobile	40%	0.1376	0.0306	2.2116	0.1776
Automobile	50%	0.1344	0.0303	2.0720	0.0485
Automobile	60%	0.1375	0.0314	2.0948	0.1680
Automobile	70%	0.1391	0.0321	2.2168	0.0876
Automobile	80%	0.1364	0.0296	2.1749	0.1293
Automobile	90%	0.1360	0.0290	2.2069	0.1498
Automobile	100%	0.1378	0.0297	2.1966	0.2062
AutoMPG	0%	33.0184	70.4411	0.8349	0.0223
AutoMPG	10%	0.1738	0.0268	1.0615	0.1267
AutoMPG	20%	0.1422	0.0213	1.1571	0.1539
AutoMPG	30%	0.1416	0.0188	1.2406	0.2088
AutoMPG	40%	0.1411	0.0195	1.2924	0.2162
AutoMPG	50%	0.1412	0.0185	1.2562	0.1616
AutoMPG	60%	0.1421	0.0193	1.2802	0.1895
AutoMPG	70%	0.1415	0.0198	1.2139	0.1292
AutoMPG	80%	0.1410	0.0191	1.2253	0.1172
AutoMPG	90%	0.1419	0.0193	1.3415	0.1811
AutoMPG	100%	0.1406	0.0195	1.3139	0.1341
Communities	0%	84.7196	194.7314	5.5254	0.7397
Communities	10%	0.3618	0.0225	6.8387	0.5967
Communities	20%	0.3576	0.0219	7.3251	0.9862
Communities	30%	0.3531	0.0217	8.8689	1.2485
Communities	40%	0.3483	0.0222	9.6139	0.4482
Communities	50%	0.3449	0.0216	11.7462	0.5614
Communities	60%	0.3432	0.0224	13.2240	1.4286
Communities	70%	0.3408	0.0220	15.0583	0.7164
Communities	80%	0.3386	0.0220	19.7774	4.2876
Communities	90%	0.3379	0.0212	33.9608	11.8286
Communities	100%	0.3367	0.0218	56.8291	14.4478

Table B.18: Results of experiment 8 for regression problems.

Binomial classification problems

Dataset	Proportion of observations used to train each SNN (EnsSNN)	Mean Accuracy	Sd Accuracy	Mean time (s)	Sd time (s)
Heart	0%	80.5556	3.7377	1.5632	0.0838
Heart	10%	78.7222	3.4441	1.7968	0.1118
Heart	20%	82.2222	3.3040	2.0580	0.3042
Heart	30%	83.0000	2.7240	2.2780	0.2955
Heart	40%	82.6667	3.0885	2.1714	0.4012
Heart	50%	83.0000	3.5339	2.0357	0.1662
Heart	60%	82.7222	3.3908	2.0879	0.3558
Heart	70%	82.6667	3.3620	2.0865	0.4299
Heart	80%	81.7778	3.5985	1.9227	0.0852
Heart	90%	82.2222	3.3538	1.9830	0.1907
Heart	100%	82.0000	3.2203	2.0476	0.1022
Pima	0%	73.6328	2.2900	1.4452	0.1295
Pima	10%	76.2891	2.3441	2.6166	0.4934
Pima	20%	76.5039	2.2107	1.9244	0.1363
Pima	30%	76.1914	1.7537	2.0830	0.1902
Pima	40%	75.5469	2.1260	2.2614	0.1935
Pima	50%	76.0742	1.7828	2.3608	0.1587
Pima	60%	75.9180	2.0249	4.4147	7.8056
Pima	70%	75.8984	1.5630	2.7588	0.1661
Pima	80%	75.5469	1.1962	3.0125	0.1925
Pima	90%	75.8203	1.0490	3.3434	0.1665
Pima	100%	76.0547	1.7763	3.8276	0.1815
HorseColic2	0%	79.5902	2.8876	2.4463	0.1893
HorseColic2	10%	81.7623	3.2120	2.7191	0.1794
HorseColic2	20%	81.8852	2.6977	2.8281	0.1917
HorseColic2	30%	82.0082	2.6121	2.8861	0.3108
HorseColic2	40%	82.1311	3.0738	2.7581	0.1756
HorseColic2	50%	82.0492	3.0426	2.6948	0.1721
HorseColic2	60%	81.9262	3.0953	2.8530	0.1630
HorseColic2	70%	82.7049	2.9287	2.7338	0.1118
HorseColic2	80%	82.9508	3.2741	2.7510	0.1390
HorseColic2	90%	82.2951	3.0474	2.8054	0.0739
HorseColic2	100%	82.4590	3.1910	2.8561	0.1182
Mammographic	0%	78.3801	2.1190	1.7418	0.3125
Mammographic	10%	79.7819	1.8369	2.3094	0.5762
Mammographic	20%	80.5296	2.2524	2.2192	0.1600
Mammographic	30%	80.5919	2.0138	2.4641	0.3959
Mammographic	40%	80.8723	1.9877	2.6607	0.2194
Mammographic	50%	80.6854	2.0082	2.9945	0.1360
Mammographic	60%	80.7477	1.7549	3.7354	0.5405
Mammographic	70%	80.8411	1.9441	3.6606	0.1788
Mammographic	80%	80.6231	1.7793	4.3401	0.1674
Mammographic	90%	80.7165	2.2001	4.9035	0.4997
Mammographic	100%	80.7788	1.9815	6.1992	1.2236

Table B.19: Results of experiment 8 for binomial classification problems.

Multinomial classification problems

Dataset	Proportion of observations used to train each SNN (EnsSNN)	Mean Accuracy	Sd Accuracy	Mean time (s)	Sd time (s)
Audiology	0%	77.8947	4.0579	2.7787	0.4967
Audiology	10%	79.6053	5.0104	2.9366	0.2906
Audiology	20%	83.0263	4.3618	2.9585	0.0933
Audiology	30%	84.8684	3.6300	3.2976	0.2498
Audiology	40%	85.6579	3.3084	3.5454	0.2066
Audiology	50%	85.3947	3.2498	3.9972	1.3920
Audiology	60%	85.2632	3.4978	5.5071	3.1191
Audiology	70%	85.2632	3.3860	3.9026	0.3461
Audiology	80%	85.9211	3.4002	4.1640	0.3743
Audiology	90%	85.7895	4.4209	4.2269	0.2923
Audiology	100%	85.9211	4.4318	4.5809	0.3197
Glass	0%	60.5556	4.4502	1.3110	0.0844
Glass	10%	61.3889	4.1821	1.9083	0.1814
Glass	20%	65.1389	5.4954	2.3678	0.1215
Glass	30%	66.2500	6.8057	2.3294	0.0657
Glass	40%	66.9444	7.5734	3.1143	0.4766
Glass	50%	68.6111	5.7153	3.8803	0.5356
Glass	60%	70.0000	7.4708	6.2264	1.4252
Glass	70%	69.7222	6.7662	11.1874	4.5028
Glass	80%	69.8611	7.4377	12.3692	1.3690
Glass	90%	70.1389	6.6205	12.9283	1.0848
Glass	100%	69.3056	6.7233	14.5146	0.9213
HorseColic1	0%	63.8843	4.6759	3.1362	0.7454
HorseColic1	10%	66.8595	4.5174	3.6907	1.0958
HorseColic1	20%	65.4545	3.9308	3.0101	0.2099
HorseColic1	30%	66.1983	3.7863	3.5724	0.1974
HorseColic1	40%	66.2810	4.7042	3.8658	0.3642
HorseColic1	50%	67.1901	4.7723	3.4967	0.2115
HorseColic1	60%	67.6860	4.5174	3.6050	0.2075
HorseColic1	70%	67.3554	5.6625	3.7723	0.2255
HorseColic1	80%	68.4298	4.8316	4.0163	0.2988
HorseColic1	90%	67.8512	4.1861	4.4014	0.5406
HorseColic1	100%	68.4298	5.1949	5.1696	1.5977
Annealing	0%	76.4662	2.2361	7.0263	3.4042
Annealing	10%	91.5414	1.7387	8.8404	2.5992
Annealing	20%	93.5714	1.0095	9.2696	1.1402
Annealing	30%	95.1880	1.0303	7.6229	3.7981
Annealing	40%	95.7895	1.0303	6.9932	0.3262
Annealing	50%	95.9774	0.7529	9.2147	0.4496
Annealing	60%	96.1278	0.8131	11.6906	0.7929
Annealing	70%	96.5038	0.7735	12.9318	0.6086
Annealing	80%	96.3534	0.9715	15.2648	0.9301
Annealing	90%	96.5789	1.0989	18.0333	1.3203
Annealing	100%	96.8421	0.9576	20.2601	1.0426
Contraceptive	0%	45.8248	1.6685	2.8132	0.2194
Contraceptive	10%	53.5031	2.3282	4.9542	0.3211
Contraceptive	20%	53.5845	1.6998	5.3912	1.0102
Contraceptive	30%	53.2383	1.7786	6.2077	0.3172
Contraceptive	40%	53.2790	2.0280	8.1833	0.7357
Contraceptive	50%	53.7882	1.9827	10.8867	0.6775
Contraceptive	60%	53.7067	2.1014	15.4909	1.3654
Contraceptive	70%	54.1752	1.9769	21.2833	3.8775
Contraceptive	80%	54.5010	2.1105	28.1963	2.5856
Contraceptive	90%	54.4603	1.7915	35.9466	5.7332
Contraceptive	100%	54.7251	1.3340	40.8154	5.7132

Table B.20: Results of experiment 8 for multinomial classification problems.

B.9 Experiment 9

Regression problems

Dataset	Percentage of missing values	Mean NRMSE	Sd NRMSE	Mean time (s)	Sd time (s)
Automobile	0%	0.1338	0.0314	2.0505	0.1258
Automobile	10%	0.1895	0.0308	2.0801	0.0957
Automobile	20%	0.2858	0.0584	2.0438	0.1354
Automobile	30%	0.3270	0.0399	2.1679	0.1517
Automobile	40%	0.3812	0.0855	2.1381	0.1238
Automobile	50%	0.5140	0.1243	2.0787	0.0461
Automobile	60%	0.5442	0.0711	2.1549	0.1676
Automobile	70%	0.5531	0.0879	2.1100	0.0868
Automobile	80%	0.6698	0.0693	2.1039	0.1680
Automobile	90%	0.8778	0.1269	2.1310	0.1010
AutoMPG	0%	0.1407	0.0187	1.1208	0.1139
AutoMPG	10%	0.1738	0.0246	1.0875	0.0770
AutoMPG	20%	0.1946	0.0255	1.1097	0.0748
AutoMPG	30%	0.2160	0.0307	1.1704	0.1419
AutoMPG	40%	0.2505	0.0306	1.1119	0.0774
AutoMPG	50%	0.3063	0.0362	1.2008	0.1953
AutoMPG	60%	0.3614	0.0421	1.2070	0.1075
AutoMPG	70%	0.4662	0.0520	1.1633	0.0936
AutoMPG	80%	0.5925	0.0553	1.2083	0.1131
AutoMPG	90%	0.7914	0.0411	1.1618	0.0959
Communities	0%	0.3456	0.0214	11.9838	1.6802
Communities	10%	0.3588	0.0191	13.9959	1.2243
Communities	20%	0.3771	0.0141	15.6987	1.2243
Communities	30%	0.3943	0.0154	17.0339	1.2090
Communities	40%	0.4180	0.0166	16.0820	0.2808
Communities	50%	0.4466	0.0208	16.0297	0.6871
Communities	60%	0.4863	0.0207	15.6502	1.6954
Communities	70%	0.5486	0.0306	14.6036	0.5575
Communities	80%	0.6262	0.0241	13.4777	0.7341
Communities	90%	0.7183	0.0172	12.2089	0.4281

Table B.21: Results of experiment 9 for regression problems.

Binomial classification problems

Dataset	Percentage of missing values	Mean Accuracy	Sd Accuracy	Mean time (s)	Sd time (s)
Heart	0%	82.5556	2.3454	2.4243	0.5888
Heart	10%	81.3333	1.8739	2.4101	0.4885
Heart	20%	80.2222	3.6590	2.0799	0.1714
Heart	30%	80.6667	3.4427	2.0429	0.1298
Heart	40%	79.8889	3.7570	2.0096	0.1839
Heart	50%	77.5556	4.4073	1.9931	0.0680
Heart	60%	76.4444	3.4665	2.0081	0.1796
Heart	70%	71.7778	3.8561	1.9767	0.0682
Heart	80%	68.3333	4.0656	1.9765	0.1373
Heart	90%	64.3333	4.6392	2.1345	0.1095
Pima	0%	76.0156	1.6388	2.2403	0.2022
Pima	10%	75.4688	1.6450	2.1874	0.1340
Pima	20%	75.5469	1.6897	2.2303	0.1639
Pima	30%	74.1406	1.5603	2.2500	0.1703
Pima	40%	73.9062	1.6346	2.1922	0.0974
Pima	50%	72.5391	2.5452	2.2094	0.1080
Pima	60%	71.4062	1.6857	2.2831	0.3597
Pima	70%	69.5312	2.0172	2.2696	0.1919
Pima	80%	68.0859	0.8037	2.4047	0.1857
Pima	90%	66.0547	2.0293	2.4163	0.2137
HorseColic2	0%	82.5410	3.7072	2.7250	0.1424
HorseColic2	10%	81.3934	3.3696	2.9410	0.1555
HorseColic2	20%	79.3443	3.7820	3.9662	1.7796
HorseColic2	30%	79.0164	3.6082	2.8241	0.3877
HorseColic2	40%	76.3934	3.4517	2.9466	0.2112
HorseColic2	50%	74.5902	4.3545	2.9595	0.5691
HorseColic2	60%	74.1803	4.7521	2.7350	0.1525
HorseColic2	70%	72.0492	5.1181	2.7345	0.1711
HorseColic2	80%	70.9016	3.0486	2.6944	0.1775
HorseColic2	90%	66.7213	2.4192	2.9629	0.1930
Mammographic	0%	80.9657	2.0998	2.9104	0.1843
Mammographic	10%	80.0935	1.9003	2.9615	0.1672
Mammographic	20%	79.0031	1.5416	2.9677	0.1856
Mammographic	30%	77.5389	2.4681	2.9592	0.1961
Mammographic	40%	75.4829	1.8227	2.9984	0.2384
Mammographic	50%	72.4299	2.0988	3.0040	0.1769
Mammographic	60%	69.9688	1.7878	3.1394	0.3481
Mammographic	70%	65.3271	2.2515	3.0333	0.1957
Mammographic	80%	60.8100	2.4032	3.0718	0.3665
Mammographic	90%	56.5732	0.8588	2.9507	0.1662

Table B.22: Results of experiment 9 for binomial classification problems.

Multinomial classification problems

Dataset	Percentage of missing values	Mean Accuracy	Sd Accuracy	Mean time (s)	Sd time (s)
Audiology	0%	85.1316	4.3882	3.1887	0.2096
Audiology	10%	84.0789	3.4227	3.2181	0.2292
Audiology	20%	82.3684	2.0758	3.1047	0.1407
Audiology	30%	77.7632	3.3661	3.1469	0.0948
Audiology	40%	75.7895	4.6911	3.1507	0.0947
Audiology	50%	73.6842	4.5580	3.1555	0.1154
Audiology	60%	72.6316	4.1050	3.1860	0.1087
Audiology	70%	67.8947	4.9699	3.3023	0.1660
Audiology	80%	63.4211	6.1654	4.2178	0.4581
Audiology	90%	49.7368	11.1786	3.4037	0.1394
Glass	0%	68.1944	6.6914	3.9979	0.6485
Glass	10%	65.2778	7.0820	3.9960	0.6591
Glass	20%	55.4167	6.5620	4.0903	0.5747
Glass	30%	52.7778	8.3590	4.2577	0.6083
Glass	40%	47.7778	4.3528	4.1174	0.4444
Glass	50%	47.3611	4.0598	4.2632	0.2267
Glass	60%	49.0278	6.1784	4.3567	0.3551
Glass	70%	39.8611	7.8853	5.2143	1.0252
Glass	80%	35.2778	9.5599	7.0388	0.3893
Glass	90%	22.2222	9.9725	4.7358	0.6540
HorseColic1	0%	67.3554	4.0347	3.4738	0.2580
HorseColic1	10%	67.5207	3.7983	3.4825	0.1954
HorseColic1	20%	66.1983	3.9433	3.3927	0.1854
HorseColic1	30%	64.3802	5.8627	3.4184	0.1872
HorseColic1	40%	64.4628	4.1960	3.5914	0.2815
HorseColic1	50%	62.3967	4.1824	3.3756	0.1515
HorseColic1	60%	62.8099	4.7874	3.2887	0.1477
HorseColic1	70%	61.9008	3.1033	3.3270	0.1547
HorseColic1	80%	60.9917	4.9861	3.4828	0.1770
HorseColic1	90%	59.0083	7.0901	3.5802	0.1660
Annealing	0%	96.2030	1.4337	8.8857	0.5814
Annealing	10%	90.9398	0.9779	9.2227	0.7101
Annealing	20%	86.4662	2.0206	9.0816	0.7699
Annealing	30%	81.4662	1.7187	9.4021	0.3634
Annealing	40%	80.6015	2.3120	9.1418	0.3654
Annealing	50%	77.5188	1.6511	8.9696	0.4294
Annealing	60%	77.2180	1.8003	8.8812	0.3040
Annealing	70%	76.2782	1.8025	10.1573	0.8900
Annealing	80%	76.5789	1.9738	11.1743	0.4686
Annealing	90%	76.0526	2.1417	9.0229	0.3572
Contraceptive	0%	53.4623	2.0236	10.9086	0.5031
Contraceptive	10%	51.1202	1.5331	9.9007	0.3630
Contraceptive	20%	49.9185	1.4448	9.8012	0.2600
Contraceptive	30%	48.1466	1.6828	9.8428	0.2592
Contraceptive	40%	46.6599	2.8293	10.2488	0.3611
Contraceptive	50%	45.4990	1.8721	10.2061	0.2406
Contraceptive	60%	44.1344	2.6660	10.9461	0.3904
Contraceptive	70%	44.8676	1.9394	10.6283	0.3213
Contraceptive	80%	43.8086	1.9308	10.3472	0.5430
Contraceptive	90%	43.5031	1.9421	9.4760	0.3775

Table B.23: Results of experiment 9 for multinomial classification problems.