FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

EDGE: Evolutionary Directed Graph Ensembles

Xavier Reis Fontes

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Daniel Castro Silva

Second Supervisor: Pedro Henriques Abreu

July 24, 2020

EDGE: Evolutionary Directed Graph Ensembles

Xavier Reis Fontes

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. Rui Camacho
External Examiner: Prof. Alberto Fernández
Supervisor: Prof. Daniel Castro Silva
Second Supervisor: Prof. Pedro Henriques Abreu

July 24, 2020

Abstract

Classification tasks are being tackled in a plethora of scientific fields, such as astronomy, finance, healthcare, human mobility, and pharmacology, to name a few. Classification is defined as a supervised learning approach that uses labeled data to assign instances to classes. A common approach to tackle these tasks are ensemble methods. These are methods that employ a set of models, instead of just one and combine the predictions of every model to obtain the prediction of the whole. Common obstacles in ensemble learning are the choice of base models to use and how best to aggregate the predictions of each individual to produce the ensemble's prediction. It is also expected to mitigate the weaknesses of its members while pooling their strengths together. It is in this context that Evolutionary Directed Graph Ensembles (EDGE) thrives.

EDGE is a machine learning tool based on social dynamics and modeling of trust in human beings using graph theory. Evolutionary Algorithms are used to evolve ensembles of models that are arranged in a directed acyclic graph structure. The connections in the graph map the trust of each node in its predecessors. The novelty in such an approach stems from the fusion of ensemble learning with graphs and evolutionary algorithms. A limitation of EDGE is that it focuses only on changing the topology of the graph ensembles, with the authors of hypothesizing about using the learned graphs for other tasks.

Our objective is to tackle the limitations of the original proposal of EDGE and bestow upon it new capabilities to improve its predictive power. This project proposes a method for updating the weights of the connections between nodes of the graph ensembles, such that the strength of the relationships between nodes evolves over time. Inspired by the notion of meta-learning, we also propose a methodology for saving learned graphs and bootstrapping different datasets with evolved graphs. We endow EDGE with bootstrapping capabilities while investigating a suitable similarity metric for dataset choice based on the extraction of meta-features from datasets.

When compared to the original EDGE, our weight evolution approach improved on 3 of the 4 datasets by an average margin of 4.20 percentage points, where the loss in the fourth dataset was of about 0.3 percentage points. Once compared with a baseline suite of models, ours achieved the best value in 34 of the 38 datasets, with gains as substantial as 30 percentage points. The bootstrap was shown to be effective in improving the prediction power, with the exploitation of previous runs improved the results on 19 out of 21 datasets.

The contributions can be summarized as a novel way to evolve graph ensembles, by also evolving the weights between nodes of the graphs, coupled with the idea of bootstrapping any dataset using previous runs from other datasets. The analysis of dataset choice for the bootstrapping lead to the proposal of a similarity metric between datasets that can be used to facilitate the choice for bootstrapping, without exhaustive or random search in the available datasets.

Keywords: Ensemble Methods, Evolutionary Algorithms, Graph Dynamics, Dataset Features, Similarity Measures

Resumo

Tarefas de classificação são abordadas numa grande variedade de campos científicos, tais como astronomia, finanças, saúde, mobilidade humana, e farmacologia, para citar alguns. A classificação é definida como uma abordagem de aprendizagem supervisionada que utiliza dados etiquetados para atribuir classes às instâncias. Normalmente, para enfrentar estas tarefas são usados métodos de *ensemble*. Estes são métodos que utilizam um conjunto de modelos, em vez de apenas um, e combinam as previsões de cada modelo para obter a previsão do todo. Obstáculos comuns no uso de *ensembles* são a escolha de modelos a utilizar nos conjuntos e a melhor forma de agregar as previsões de cada indivíduo para produzir a previsão do *ensemble*. Também se espera a atenuação das fraquezas dos seus membros, ao mesmo tempo que se reunem os seus pontos fortes. É neste contexto que o Evolutionary Directed Graph Ensembles (EDGE) prospera.

EDGE é uma ferramenta de aprendizagem automática que usa algoritmos evolucionários para desenvolver conjuntos de modelos que estão dispostos em grafos acíclicos dirigidos. As ligações no grafo mapeiam a confiança de cada nó nos seus predecessores. A novidade em tal abordagem deriva da fusão de aprendizagem por *ensemble* com grafos e algoritmos evolucionários. Uma limitação do EDGE é que se concentra apenas na alteração da topologia dos *ensembles* em grafo.

O nosso objectivo é resolver as limitações da proposta original do EDGE e dotá-lo de novas capacidades para melhorar o seu poder de previsão. É proposto um método para actualizar os pesos das ligações entre os nós dos *ensembles* em grafo, de modo a que a força das relações entre os nós evolua ao longo do tempo. Inspirados pela noção de meta-aprendizagem, propomos também uma metodologia para guardar os grafos aprendidos e para inicializar diferentes conjuntos de dados com grafos evoluídos. Dotamos o EDGE de capacidades de bootstrapping enquanto investigamos uma métrica de similaridade, baseada na extracção de meta-características, para a escolha do conjunto de dados a servir de inicialização. Quando comparado com o EDGE original em termos de *Accuracy*, a nossa evolução de pesos melhorou em 3 dos 4 conjuntos de dados com uma margem média de 4,20 pontos percentuais, onde a perda no quarto conjunto de dados foi de cerca de 0,3 pontos percentuais. Uma vez comparado com um conjunto de modelos *baseline*, o nosso método alcançou o melhor valor em 34 dos 38 conjuntos de dados, com ganhos tão substanciais como 30 pontos percentuais. O uso de bootstrap provou ser eficaz na melhoria do poder de previsão, com a exploração das execuções anteriores a melhorar os resultados em 19 dos 21 conjuntos de dados.

As contribuições são resumidas como uma nova forma de evoluir os *ensembles* em grafo, evoluindo também os pesos entre os nós dos grafos, juntamente com a ideia de bootstrap utilizando execuções anteriores noutros conjuntos de dados. A análise da escolha do conjunto de dados para o bootstrapping leva à proposta de uma métrica de semelhança que pode ser utilizada em vez de uma pesquisa exaustiva nos conjuntos de dados disponíveis.

Keywords: Aprendizagem por *Ensemble*, Algoritmos Evolucionários, Dinâmica de Grafos, Características de Conjuntos de Dados, Métricas de Similaridade

Acknowledgements

None of this work would have been possible without all the intellectual, emotional and unconditional support I received from my teachers, friends, and family. Throughout this experience, I was able to grow, be more critic of my work, and hopefully, plant some seeds for the future.

Xavier Fontes

"Who makes the world?
Perhaps the world is not made.
Perhaps nothing is made.
Perhaps it simply is, has been, will always be there...
a clock without a craftsman."

Alan Moore

Contents

List of Figures

List of Tables

Abbreviations

DAG	Directed Acyclic Graph
DB	Database
DGE	Directed Graph Ensemble
DM	Distance Metric
DS	Dataset
DT	Decision Tree
EA	Evolutionary Algorithm
EDGE	Evolutionary Directed Graph Ensembles
EoG	Evolution of Graphs
FM	Feature Manipulation
GA	Genetic Algorithm
GB	Gradient Boosting
KS	Kolmogorov–Smirnov
MC	Model Choice
MF	Meta-Feature
ML	Machine Learning
PD	Promoting Diversity
PoDAG	Partition of Directed Acyclic Graphs
PoG	Partition of Graphs
RF	Random Forest
SGD	Stochastic Gradient Descent
SM	Similarity Measure

Chapter 1

Introduction

Machine Learning tasks such as Classification and Regression have become ubiquitous in most problem domains [Müller et al., 2016]. One group of methods that are considered robust and provide excellent results are ensemble type models [Meir and Rätsch, 2003]. These qualities stem from the aggregation of different models that together mitigate each member's shortcomings while building upon their strengths. Nonetheless, there are still problems with ensembles that have become widely studied areas of research, such as the choice of which models should incorporate a given ensemble and how to combine the predictions of many models into one.

1.1 Context and Motivation

EDGE is an ensemble type method that evolves a population of ensembles where each ensemble is represented as a directed graph with the nodes of the graph being standalone models [Fontes and Silva, 2019]. This evolution of ensembles explores the problem of which models to include by random picking the models and let the population of ensembles be evaluated and evolved, over time. The combination of predictions is made such that each node's prediction is combined with the prediction of its predecessors and passed to its successors in the graph.

Since we employ the metaphor of trust dynamics in the human population, we would like the models in the ensemble to be connected with one another, hence the graph. It's desired that one's prediction can influence the prediction of another, hence the direction of trust/predictions from one node to another, justifying the directed part in directed graph ensembles. Having cycles in the graph might lead to ambiguities in how the predictions flow from one another, hence the representation of the graph ensembles as directed acyclic graphs (DAGs). Of course, the acyclic property is not present in the dynamics of human populations; nonetheless, we argue that it better suits the general purpose of EDGE, to improve classification power.

EDGE had some shortcomings that characterize the stage for this work. It only evolved the topology of the ensembles during the evolution, and not a lot of testing was done to compare it to

baselines. The authors point to further directions being in developing a way to evolve the weights of the graph ensembles and briefly discuss the flexibility and customization ability as desiderata for EDGE.

We are motivated by the idea that ensembles can be formed in a way that each node, taking into account the predictions of other nodes, can lead to a more robust prediction tool than traditional ensembles. The idea of making use of previously evolved graphs is also a strong incentive to delve into EDGE further. In a more holistic sense, we believe this can be a reliable tool to be used by many colleagues, allowing for the sharing of exciting and unique ideas on how the population and its individuals can be evolved within the EDGE framework.

1.2 Objectives

The main goal of this thesis is to improve the predictive power of EDGE. This entails several sub-goals. Particularly, dealing with some shortcomings of EDGE, while introducing novel developments to the idea of evolving graph ensembles. Another sub-goal is also the preparation of baseline tests in order to compare the developments in a standard way.

One principal direction will be the evolution of the weights in the graph ensembles, as well as the weight each node gives to its prediction. Another point of focus is the bootstrapping of EDGE with learned graphs while studying possible similarity measures to compare distinct datasets.

The following research questions will be answered by the end of this document. We express them here, in a succinct manner:

- *Can EDGE's performance be improved by updating the weights of the graph ensembles?*
- *Can EDGE's performance be improved by using graphs learned from other data?*
- *What is the relation between EDGE's configuration and its performance?*
- *Can a similarity metric be proposed that accurately chooses the best data to bootstrap EDGE with?*

1.3 Methodology and Expected Results

In order to answer the research questions posed in Section 1.2, the first duty is that of rebuilding and adapting the code base for more comfortable usage, allowing for quick prototyping and implementation of new features, as well as a soft learning curve for distribution among other people. The work tackles the weight evolution of the graphs. We formulate such a problem as an optimization problem, where we want to optimize the weights between nodes of the graphs and the weight each node gives to its prediction, such that the ensemble as a whole has its prediction power maximized. In order to balance the weight evolution together with the topology change, we introduce a decision mechanism to switch between weight updating and topology updating, whenever it deems it favorable. In order to make EDGE take advantage of graphs that have been

evolved for other datasets, we will implement a feature for saving the best performing graphs, and corresponding base models, in a database. This feature, coupled with the features for reading the database and updating EDGE's graph initialization process and base model selection process, makes it possible for EDGE to bootstrap. We consider this to be a task of meta-learning, where we use what was learned in one dataset, to achieve better results, faster, in a different dataset. The details of these proposals are presented in Chapter 4.

In order to understand the relation between EDGE's performance and some of its parameters, we opt to test our proposals in a comprehensive suite of datasets. These datasets are further described in Chapter 6. Likewise, the similarity metric proposed is based on the extraction of meta-features of the datasets. Meta-features are general properties that are capable of characterizing datasets. We refer the reader to a more in-depth explanation of meta-features in Section 2.6. The similarity measure quantifies how similar two datasets are, independently of their size, number of features, types of features, and the number of classes to be predicted.

To answer the research questions with more confidence, a baseline suite of models was designed and tested on the same datasets, to be able to directly compare EDGE with the types of models that EDGE itself uses as base models. Said baseline suite is also described in Chapter 6. The results from all the experiments closely match our expectations that weight updating indeed improves on the classification performance, as well as the bootstrapping mechanism. The analysis of the results also indicates that we can choose a similarity metric such that, in a general way, one of the best datasets is used to bootstrap any other. Two scientific articles originated from our work, one accepted for publication [Fontes et al., 2020] and another in development.

1.4 Document Structure

The rest of this document is organized as follows. In Chapter 2, we provide the reader with a brief description of key concepts and terminology needed to understand our contribution in its entirety. Chapter 3 details relevant and related works to our domain of research, as well as an analysis of where our work sits concerning other works. With Chapter 4, we detail the problem where our focus is aimed and provide an overview of the designed solution. In Chapter 5, we expose technical decisions and trade-offs about the implementation of the solution. In Chapter 6, we mention the experiments that were performed, alongside the corresponding discussions for each of the experiments. Finally, we summarize the work performed and future directions that we have considered, in Chapter 7.

Chapter 2

Background Knowledge

In this chapter, we introduce the fundamental concepts needed to grasp this thesis' work. Our descriptive journey starts at the topics that are the basis for EDGE, i.e., Evolutionary Algorithms, Ensembles and Graph Theory, in Sections 2.1, 2.2 and 2.3. We proceed to acquaint the reader with EDGE, in Section 2.4. Next, focusing on our intended proposals, we will review some notions in Meta-Learning, Meta-Features of data and Similarity Measures, in Sections 2.5, 2.6 and 2.7. Our focus in each section will be to provide the necessary information while sticking to the scope of this work.

2.1 Evolutionary Algorithms

Evolutionary Algorithms take inspiration in Biology and the evolution of species [Darwin, 2009]. These algorithms refer to a large field of global optimization methods. Conceptually, Evolutionary Algorithms start with a random population and incrementally generate new, better populations, where each individual in the population is a candidate solution to the problem [Yu and Gen, 2010].

Each individual is encoded as a set of characteristics or genes. These are smaller components that together define the behavior or performance of the individuals. The genes are manipulated to obtain new individuals, and their manipulation depends on how the individuals of a population are represented, gene-wise.

In Fig. 2.1, we have a generic example of such algorithms work. Observe the step of evaluating the fitness of the population. *Fitness* is a measure that allows us to evaluate each individual, in a quantitative form, concerning each other. The fitness function is the function we are trying to optimize, either maximize or minimize without loss of generality. Considering a fitness function $Fitness(X)$, then all individuals, when evaluated are assigned a number such that, for two individuals X_1 and X_2, if $Fitness(X_1) > Fitness(X_2)$, then individual X_1 is better than X_2.

After evaluating the population, we generate a new population. For the context of this work, we consider the generation of new populations using evolutionary operators, from a sub-field of

Evolutionary Algorithms, Genetic Algorithms. The most common operators are*selection*, *mutation* and *crossover* [Bäck et al., 2000]. It is also frequent the use of *elitism* in evolving populations. We refer to Fig. 2.1, the logic for evolving a population, this time using some of the operators mentioned.

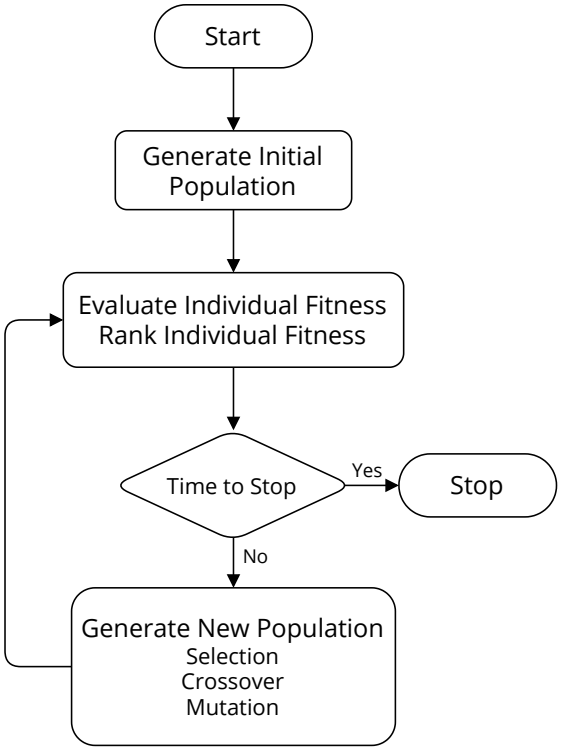

Figure 2.1: Flowchart of Genetic Algorithm (adapted from [Kachitvichyanukul, 2012])

Elitism is applied by selecting the top-N best individuals to be present in the next generation, without regards to the other operators. The *mutation* operator introduces variability in individuals by changing some of its genes. The variability introduced allows the population as a whole to escape local optima and to discover possibly useful genes that are later passed to the following generations. It is also desirable to combine individuals such that new individuals can be produced that aggregate the best genes of each of its parent individuals. This operator is commonly referred to as *Recombination* or *Crossover*, where two individuals are crossed, and one (or more) new individual(s) take their place. The *selection* of which individuals are crossed usually depends on how good of a fitness value they have. For example, individuals with better fitness have higher chances of being selected for the crossover, and thus continue the process of evolution, than individuals with worse fitness values [Bäck et al., 2018].

2.2 Ensembles

The idea behind ensemble learning is to use several models, train them on available data, and combine their predictions in order to achieve better predictive performance, as well as robustness to

possible weak performing models [Opitz and Maclin, 1999]. The robustness to weak performance stems from the use of different models, whether by using conceptually different models or train the instances of the same model in different subsets of the data.

From a holistic point of view, we have two focus centers when it comes to ensemble learning, the *data* and the *models* perspective [Géron, 2019]. By *data*, we mean that we take different subsets of data or different features of data for each model, such that any single model specializes only over a part of the whole data. This exploitation of subsets of data also merits discussion. There are several ways to divide data, but the two most common are sample-wise subsets and feature-wise subsets. In most problems, both are used together to produce different models for the ensemble. In a *model*-centric perspective, the way to construct the base models is from conceptually different sources. The assumption is that if a model has its learning concept different from another, then it will make use of data differently. Combining conceptually different models allows us to strengthen the outcome with models that are best equipped for certain types of problems while mitigating possible shortcomings of other, less adequate models [Zhang and Ma, 2012].

Two general approaches in ensemble learning are to aggregate all the base models' results, usually referred to as *Bagging* or an incremental approach where the ensemble's performance is improved step by step, commonly referred to as *Boosting*.

2.3 Graph Theory

We define a graph as a mathematical structure $G(N,E)$, where N is a set of nodes and E a set of edges, such that each edge connects two nodes from N [Wilson, 2010]. Edges can have one or more associated weights that quantify a given relationship between the connected nodes. For our case, we consider only directed graphs, meaning that each edge has a direction from node X to Y, henceforth described as E_X^Y.

Directed Acyclic Graphs (DAGs) are a subset of directed graphs that, as the name suggests, contain no cycles in their topology. A cycle in a graph can be defined as the existence of any *non-zero-length path*, following the directed edges available, that allows starting at a node X and return to X [Thulasiraman and Swamy, 2011]. An example of a directed acyclic graph is presented in Fig. 2.2

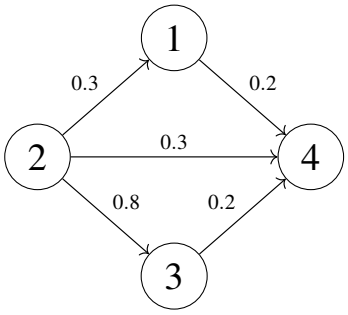

Figure 2.2: Directed Acyclic Graph Example

When dealing with DAGs, we can also define the concept of source and sink nodes. Source nodes are those that have no incoming edges, while sink nodes are defined as not having outgoing edges. In Fig. 2.2, node *2* is a source, while node *4* is a sink.

2.4 EDGE

As the name suggests, EDGE (Evolutionary Directed Graph Ensembles) is a technique that evolves ensembles. EDGE uses evolutionary algorithms to evolve a particular type of ensembles, Directed Graph Ensembles (DGEs). The novelty lies in the fact that the ensembles are represented as directed graphs. In a DGE, each node represents a standalone model, a Component Model (CM), and the connections between CMs can be thought of as encoding a measurement of *trust* between nodes.

The CMs are randomly sampled from the *Reservoir*, a structure that acts as a resource pool for all types of model-configuration pairs that can be used. These configurations have default values but are also one of the factors of EDGE that can be customized. A brief diagram of EDGE is presented in Fig. 2.3. The intuition behind the DGEs is that each node can weigh its prediction together with the aggregated prediction of its predecessors, to provide a more informed response. Since EDGE evolves the graph ensembles, as well as each of the CMs, we found it useful to illustrate how data is used in a typical experiment using EDGE, in Fig. 2.4.

As described in Section 2.2, ensembles aggregate several base models. With EDGE, each base model is a node in the graph ensemble. Each directed edge E from X to Y, E_X^Y, denotes that X trusts Y with a certain weight W_X^Y. The constraint of a graph being directed and acyclic allows us to compute the *flow* of predictions more easily across all nodes, starting in the nodes that have no incoming edges (do not trust any other node), ending at a designated *sink* node. This node outputs the final prediction of the ensemble as a whole. In a general way, the prediction of node i, P_i, for a given data point X_j, can be computed as follows:

$$P_i(X_j) = \alpha \times sp_i(X_j) + (1-\alpha) \times \frac{pp_i(X_j)}{\|pp_i(X_j)\|_1}$$

Where

$$pp_i(X) = \sum_{n \in predecessors(i)} \frac{P_n(X) \circ W_i^n}{\|P_n(X) \circ W_i^n\|_1}$$

Considering that:

- $\circ$ represents the Hadamard product;
- $sp_i(X)$ denotes the prediction made by CM_i;
- $pp_i(X)$ represents the aggregated prediction made by the predecessors of i;
- W_i^n represents the trust vector of node i in its predecessor, node n.

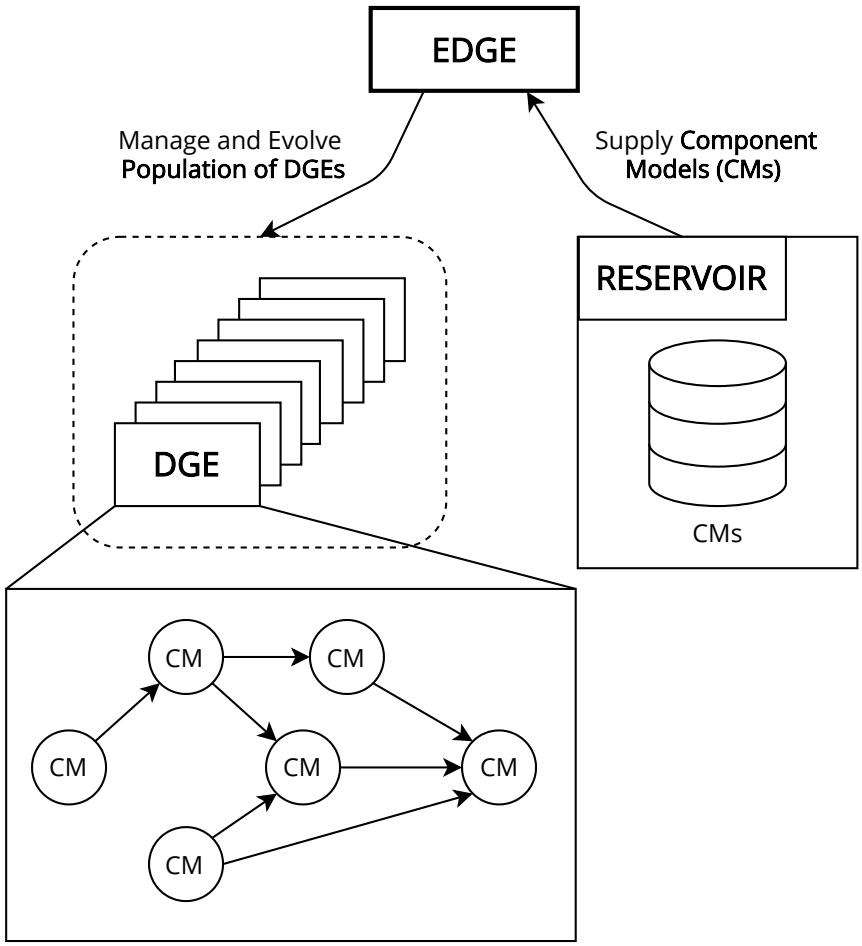

Figure 2.3: Architectural Diagram of EDGE

An illustration of EDGE's working can be seen starting the population of graph ensembles with DGEs similar to Fig. 2.5, and after a set of evolutionary steps, the results are DGEs like the example of Fig. 2.6.

EDGE was tested on 4 different datasets, the description of which will be discussed in Section 6.1, and the results we extracted are summarized in Table 2.1. This summary table aggregates the best results of two experiments where EDGE used a *Reservoir* with only Decision Trees, and another with Decision Trees, Random Forests, and Gradient Boosting Classifiers.

Table 2.1: Previous results from EDGE. Results from EDGE are the best results presented in [Fontes and Silva, 2019]

	EDGE	
Dataset	Accuracy	F1-Score
MNIST	97.78	97.78
Anuran	99.17	99.16
Appliances	86.27	84.29
Parking Lot	87.68	87.10

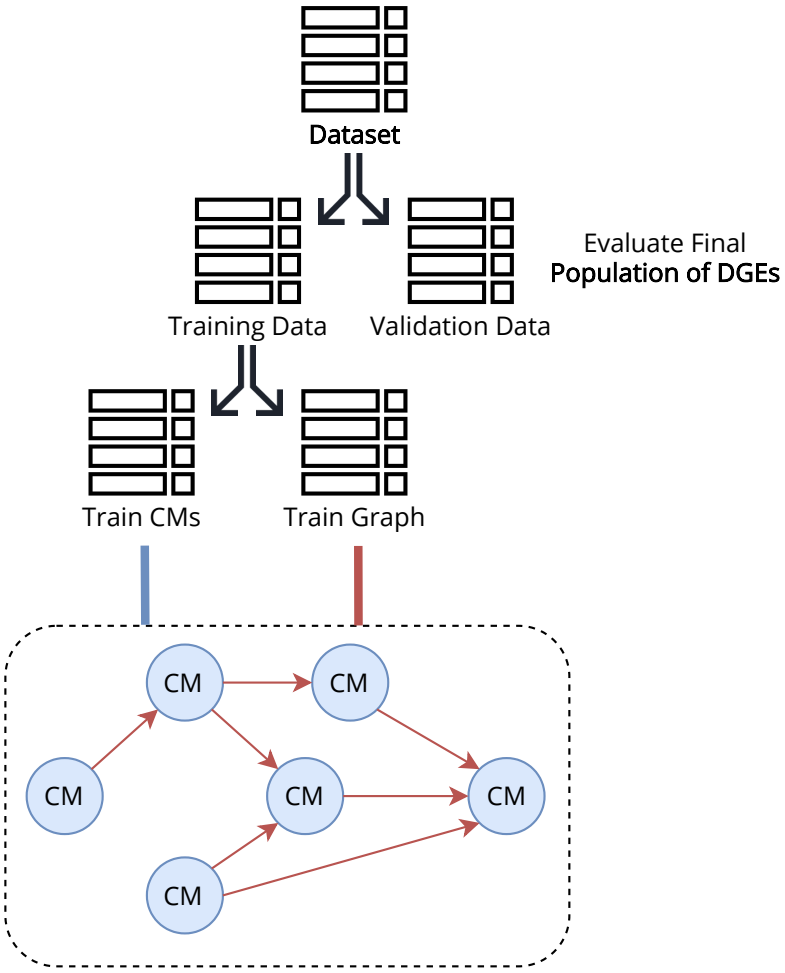

Figure 2.4: Data Flow in EDGE

2.5 Meta-Learning

Meta-Learning is a widely studied research topic and somewhat challenging to define in exact terms. However, it revolves around the idea to answer the question of how can knowledge about learning itself be exploited, such that we can produce stronger learners [Vilalta and Drissi, 2002]. Meta-learning is usually associated with hyperparameter optimization, such that systems are designed in a way that can learn the best parameter values for their own configuration.

Within this thesis, we will use the term *bootstrap* to refer to a domain of meta-learning where we consider learning as something that does not need to start from scratch with each new task.

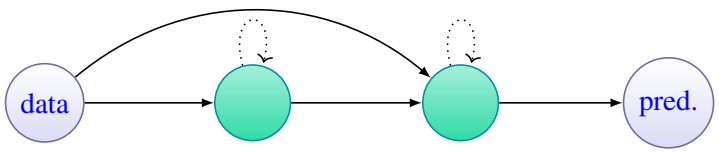

Figure 2.5: DGE Example (adapted from [Fontes and Silva, 2019])

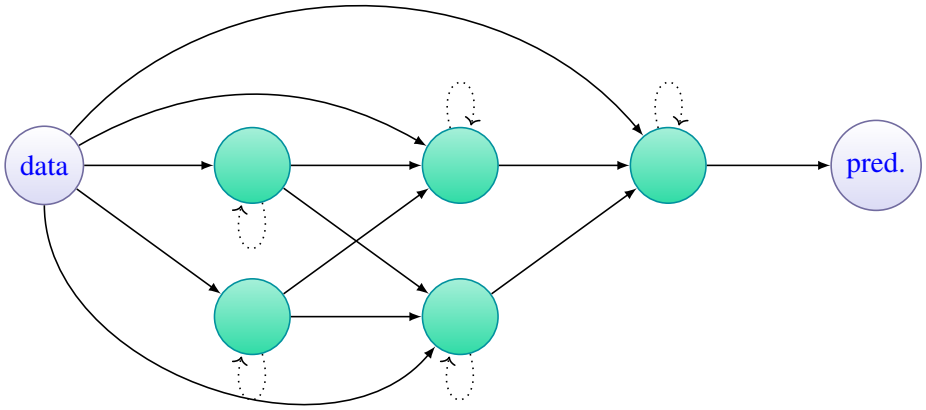

Figure 2.6: DGE at end of Evolution (adapted from [Fontes and Silva, 2019])

Instead, we can utilize current knowledge in different tasks to improve on our knowledge acquisition in other, unseen tasks. In our context, a task is a dataset which we are trying to predict, as accurately as possible.

There is also a research area that is designated as *transfer-learning*. While our proposal that is depicted in later chapters emphasizes the learning in one dataset to bootstrap another, we do not consider it to be in the realm of transfer-learning since the learning that occurs in one dataset is not explicitly designed to improve the performance on another. Nonetheless, to avoid confusion, we refer to meta-learning as the topic where our proposal is inserted and use bootstrap as the mechanism through which we will endow meta-learning capabilities to EDGE.

2.6 Meta-Features of Datasets

In the context of supervised classification, a dataset is a collection of data points from which we want to predict a given target class y, based on a set of input features x. The whole dataset can thus be represented as $X = [x_1, x_2, ..., x_N], Y = [y_1, y_2, ..., y_N]$, where each $x_i = [f_1, f_2, ..., f_M]$, with N being the number of data points, M being the number of features, and f_j being the value of feature j. Common terminology states that each dimension of x_i is a feature of the dataset.

Most of the time, it is impossible to compare a feature from dataset A to another feature of dataset B, since they can be data that has been extracted from a plethora of different domains and situations. Nonetheless, some attributes that represent a trait of the dataset as a whole can be computed. These traits are what can be described as *meta-features*. Meta because it is not a direct feature of the dataset but rather a characteristic extracted from a given dataset as a whole [Rivolli et al., 2018]. General examples of simple meta-features are, for instance: the number of data points, the number of features in the data, the number of target classes in the dataset, and the types of features present in the data.

Within the scope of this work, we take a look at more sophisticated meta-features, based in two key domains, landmark, and complexity. The reason for this choice is that they are both domains of meta-features that can be used in virtually all datasets, without enforcing restrictions as to the kind of data each dataset has.

We look at meta-features that are referred to as being in the *landmark* domain. Such measurements are based on the performance of simple and efficient models. Examples include the accuracy of a Naive Bayes classifier or a pruned decision tree. The extraction of these features is straightforward since we need only to fit the model and score its predictions.

Another domain where we focus our attention on meta-feature extraction is that of *complexity* measures. There are several different complexity measures, but the general idea is to quantify how easy it is to separate the points of the dataset in its corresponding classes. Some examples are the computation of the maximum discriminative power of the features of the dataset, the average number of points per dimension, and the error rate of linear classifiers.

2.7 Similarity Measures

We define a similarity measure as a function that takes two objects and returns a real value that expresses how similar the two objects are [Choi et al., 2010]. It is also worth mentioning that both similarity and dissimilarity measures are explored to compare the similarity or differences between two objects, respectively [Goshtasby, 2012].

From a similarity measure, we can attain resembling dissimilarity measures and vice-versa. Distance metrics are also commonly referred to in the context of evaluating how similar two objects are by considering how *close* they are to each other, in a given N-Dimensional space, which is an entire research field in and of itself [Yang and Jin, 2006]. As an example, given a distance metric between two points $Dist(X,Y)$, we can define a similarity measure for points as:

$$Similarity(X,Y) = \frac{1}{Dist(X,Y)^2 + 1}$$

The squaring and the addition are there for mathematical convenience to avoid dividing by zero, but otherwise, it is a straightforward modification.

There are copious applications for similarity metrics [Cha, 2007], but our focus will be on studying and establishing such types of metrics between datasets. This is no trivial matter because datasets can have different numbers of samples, features, and even different types of features.

Chapter 3

Literature Review

The work of this dissertation, improving and developing EDGE into a full-fledged framework, entails several research fields. EDGE evolves directed graph ensembles using evolutionary algorithms, as such we describe Evolutionary Algorithms, Ensembles and Graph Theory in Sections 3.1, 3.2 and 3.3, respectively. We intend to search for patterns in the learning phase of EDGE and understand if we can bootstrap EDGE with graph ensembles that have achieved good results in similar datasets. Similarity is covered in Section 3.4. Meta-learning, within the scope of this work, is covered in Section 3.5. A conclusion to this chapter is then presented in Section 3.6, together with the justification for our work.

3.1 Evolutionary Algorithms

Evolutionary Algorithms are primarily used in optimization problems, more specifically, multiobjective optimization. Multiobjective optimization means our objective is to optimize more than one objective function with regards that different objective functions might impede each other. Nonetheless, a solution that satisfies all objectives, as good as possible, is desirable, and these types of algorithms lead to several non-dominated solutions [Antonio and Coello, 2018].

We often talk about Evolutionary Algorithms, but this field is populated by three main streams of research, Evolutionary Programming, Genetic Algorithms, and Evolution Strategies [Bäck et al., 1993, Bäck, 1996]. All of these streams of research are similar in the sense that they are based on optimization through the concept of evolution. The crux is the start of the evolution with a population of random individuals that are iteratively evolved towards better individuals.

From a technical point of view, Evolutionary approaches have also been thoroughly tested against each other in order to understand if any provides a generally good approach, albeit restricted to the No Free Lunch Theorem [Wolpert et al., 1997]. The conventional approach for comparison is to have complex functions that are optimized in order to understand how fast and how good the solutions found by the algorithms are [Zitzler et al., 2000, Elbeltagi et al., 2005].

Both comparison works [Zitzler et al., 2000, Elbeltagi et al., 2005] allow us to understand that while there is not a general-purpose algorithm, there are algorithms such as Particle Swarm Optimization that perform satisfyingly good across many problems. It is also essential to understand that since many different variants exist, for each problem, there needs to be work made in the direction of understanding how to represent the individuals of a population best, so that evolutionary algorithms can produce the best possible results.

There is much to be discussed when talking about evolutionary approaches to ensemble learning. It fits right in the category of this thesis since, while we evolve specific types of ensembles, the concept of evolving ensembles of models has already been researched. Different approaches for the evolution of ensembles have been proposed, such as the evolution of the entire ensemble [Kim et al., 2002, Moyano et al., 2019], the use of evolutionary algorithms to choose how the data is fed to the ensemble [Wang and Wang, 2006, Galar et al., 2013] and even incremental improvements of existing ensembles [Onan et al., 2017].

Evolutionary Algorithms are used in a meta-learning fashion to present Meta-Evolutionary Ensembles [Kim et al., 2002]. Evolutionary Algorithms evolve ensembles and base classifiers such that ensembles are rewarded for choosing the best classifiers, and the base classifiers are rewarded based on their performance on more challenging data points. The resulting ensemble improves on the classification acumen of individual methods while rivaling other ensemble approaches with a smaller ensemble size. Smaller ensemble size is also desirable since it is correlated with computational costs.

Weighted sample points are used together with a Genetic Algorithm incorporated to search the weighting space in order to build more robust and accurate ensembles in work presented in [Wang and Wang, 2006]. The experiments were compared to two conventional ensemble-learning approaches, AdaBoost and Bagging, in the problem of face detection. The results showed that the proposed evolved ensembles performed on par with the other approaches but more effectively balancing the accuracy and diversity of its base classifiers. Diversity is essential in ensemble building since the focus on diversity favors the exploration of as many different patterns in data as possible.

A common way to tackle imbalanced datasets is to diminish the number of the dominant class by only keeping a smaller subset of examples, without overtaking less frequent classes. Enhancing ensembles to deal with imbalanced data through evolutionary undersample, EUSBoost was proposed [Galar et al., 2013]. EUSBoost constructs an ensemble using a Boosting algorithm but undersampling the majority class instances used to train the different base classifiers. Results of experiments showed that EUSBoost, when configured to promote diversity of classifiers, achieves better results than other approaches designed for data imbalance. Statistical tests were presented that supported the hypothesis of EUSBoost having better accuracy than other methods, for all except one.

A model-centric approach to evolving ensembles is to have a population of base classifiers that are trained and evolved such that only the best base classifiers keep being trained, leading the population as a whole to evolve towards improving the output of the ensemble. An example of this is produced in [Ai et al., 2019], where the base classifiers are neural networks with varying

configuration and size parameters. The approach was used to optimize power demand prediction of households and was shown to better predict single household power demands and have increased stability comparing with single network methods.

Within the context of graphs, the most common is the application of Evolutionary Algorithms in problems that are fundamentally encoded as graphs, but whose adequate representation is made in order to represent individuals as candidate solutions [Bui and Moon, 1996, Fleurent and Ferland, 1996]. The idea of evolving topologies has been used in [Stanley and Miikkulainen, 2002] with the evolution of neural networks through a method called NeuroEvolution of Augmented Topologies (NEAT). The results on two known learning problems, Pole Balancing and Double Pole Balancing, showed that this method of evolving neural networks managed to evolve the structure when needed with better scores and smaller solutions than other methods. NEAT is also robust to local solutions.

Evolutionary machine learning can be applied to a plethora of problem domains. The Evolutionary ensemble learning is one of the emerging topics according to a survey of 2019 [Al-Sahaf et al., 2019]. As such, we can expect new developments in this field with new ways to use evolutionary algorithms to our advantage.

We present a summary of our findings in Table 3.1.

Work	E. Ensemble	E. Data	E. Networks
[Stanley and Miikkulainen, 2002]			X
[Kim et al., 2002]	X	X	
[Wang and Wang, 2006]		X	
[Galar et al., 2013]		X	
[Moyano et al., 2019]	X	X	
[Ai et al., 2019]	X		X

Table 3.1: Layout of Evolutionary work on Ensembles, Data and Networks. E (Evolving).

3.2 Ensembles

In Section 3.1, we discuss works that have used evolutionary algorithms to evolve ensembles. As such, in this section, we take particular interest in ensemble construction and present the multitude of domains where they have been applied to.

Ensembles use more than one base model intending to attain more prediction power combining the predictions of several models. Another goal is that of robustness, meaning that the effect of especially bad base models will be mitigated. Two of the most common branches in Ensemble Learning are learning through Bagging and learning through Boosting [Alfaro et al., 2018]. We can think of Bagging as pulling all the predictions of the base models together, whether it be by simple averaging, weighed averaged, and others, in order to reduce the variance of our outcome. Boosting focuses on improving the performance of weak learners, meaning that we incrementally become better with each base model added [Alfaro et al., 2018].

Typical examples of Bagging and Boosting are Random Forests and Gradient Boosting, respectively. Random Forests attempt to average out possible mistakes by building an extensive collection of de-correlated trees [Hastie et al., 2009]. Gradient Boosting relies on minimizing a loss function with the sequential addition of models and their respective contributions to the final prediction [Breiman, 1997]. While ensemble construction is divided in this way, there are further actions we can take that impact prediction power and have been shown to be a decisive step in ensemble learning.

When constructing ensembles, we can enforce a certain diversity of the models. The idea is that fundamentally different models will learn fundamentally different patterns in data. The result is that the more patterns in data, the better we can expect to predict, assuming the underlying phenomena and structure is the same. This is commonly called model selection [Seni and Elder, 2010]. If we consider different models trained on different data, then we enter the realm of data manipulations. Data manipulations are a class of manipulations that can also be done in ensemble learning. Examples of the most common methods are feature selection and subsampling of the training data. Feature selection allows us to train each base model on a subset of all the features of the complete dataset. This training procedure leads to models that are inherently different since they have been trained on different data. Following the same idea that different data produce different models, we can assign each model to a subset of the training data [Mendes-Moreira et al., 2012]. It is worth noting that when we mention a subset of the training data or a subset of features, we can, and most times have, overlapping sets. The idea to use subsets of data or features is also one of removing possible redundancy, leading to simpler models and less computational costs, effectively learning the same with fewer data.

Another aspect of ensemble learning is the concept of pruning. In the training process, we can have base models that fit the training data entirely or fit the data in such a way that we consider the model to be *overfitted*. This fitting of the data means that such a model will typically have poor generalization properties, which is most undesirable since we are looking for a model that can learn from the data in order to predict samples which it has not seen before [Runkler, 2016]. In Random Forests, the trees that make up the forest are not pruned, since possible overfitting is mitigated by all other trees that take into account different features in the data [Zhang and Ma, 2012]. On the other hand, a standard boosting algorithm, based on Gradient Boosting, called eXtreme Gradient Boosting (XGBoost), uses pruning [Chen and Guestrin, 2016] in its trees.

Following the idea of diversity concerning the models used, the authors of [Lertampaiporn et al., 2012] used typical features of protein properties. Further, they suggested their own, more discriminative, features together with heterogeneous models to improve the classification of microRNA proteins. More than feature selection, new features were produced, and the results showed that not only was the ensemble better than single-classifier methods, the use of more discriminative feature improved classification performance. This idea of different models through feature manipulation can be contrasted with the work of social media categorization in [De la Peña Sarracén, 2017], where the authors experimented with ensembles composing of different types of models, such as Logistic Regression, Support Vector Machine, Naive Bayes classifier and K-Nearest

Neighbor for classification of Twitter[1] messages. The results showed that the ensemble of different models performed better than any standalone model, achieving a higher F1 score. It is worth mentioning that this ensemble used a weighted average for the computation of the ensemble's prediction.

It has also been shown that ensemble learning can be improving by combining different common strategies in ensembles. The authors of [Webb and Zheng, 2004] argued that more diverse ensembles could be created if different techniques were combined to provide more accurate ensembles, at a possible cost of individual classifier accuracy. The experiments were made with several combinations of techniques, mainly decision trees created with the C4.5 algorithm, trees from C4.5 with Stochastic Attribute Selection, Boosted ensembles, MultiBoosted Ensembles, and AdaBoost ensembles.

We would be remiss if we did not mention the work proposed in [Zhou et al., 2018], where an ensemble of models was connected to data points using graph theory, more specifically, the stable marriage problem [McVitie and Wilson, 1971] between models and data. The authors formulated a fitness function based on how big the subset of the dataset each model had access to, how many models were connected to each data point, and introduced a term to promote model diversity. The problem was tackled using evolutionary algorithms in order to navigate the solution space towards an adequate arrangement such that the final ensemble achieved better results in datasets of image recognition such as CIFAR10, CIFAR100, Fashion-MNIST, and STL while maintaining competitive efficiency.

With such an embracing topic, we must refer the interested reader to the following works that accurately portray the fundamentals and details of ensemble learning [Dietterich, 2000, Oza and Russell, 2001, Polikar, 2012, Zhou, 2015], and present a summary of our findings in Table 3.2.

Work	Diversity through FM	Diversity through MC
[Webb and Zheng, 2004]	X	X
[Lertampaiporn et al., 2012]	X	
[De la Peña Sarracén, 2017]		X
[Zhou et al., 2018]	X	X

Table 3.2: Layout of Ensemble works on Feature Manipulation (FM) and Model Choice (MC).

3.3 Graph Theory

Within the domain of machine learning, graphs are commonly associated with representations of knowledge or information, to find patterns or extract more facts about the represented domain [Nickel et al., 2016]. Since EDGE makes use of graphs, not to represent a vast knowledge base, but instead as a chief component in how the ensembles establish themselves, we focus our

[1] https://twitter.com

attention on the topic of graph partitioning, with emphasis on directed acyclic graphs and possible graph evolution algorithms.

Using evolutionary algorithms, the authors of [Tettamanzi, 1998] proposed a way to evolve partially connected graphs. Encoding the graph as a sequence of pairs of nodes the size of all available nodes, such that two successive nodes can be used to infer an edge between them. Two restrictions imposed were that the number of times two edges crossed each other should be as small as possible and that any edge length was as close as possible to a parameter L. An interesting take from this work is that as the graph is encoded as a sequence of nodes, crossover operations and mutations become easier to conceptualize.

The relevance of graph partitioning, specifically directed acyclic graph partitioning, is that it would allow for a relevant crossover operator to be defined without relying on some encoding of graphs that might not capture the importance of nodes connected between each other. To this end, we explore several works that dealt with graph partitioning [Hendrickson and Kolda, 2000, Patwary et al., 2019], specifically the case of directed acyclic graphs [Alamdari and Mehrabian, 2012, Herrmann et al., 2017, Moreira et al., 2018]. We also studied possible ways to evolve graphs over time, searching support for our ideas about graph mutation, in [Lieberman et al., 2005].

Graphs can also be used to describe complicated calculations, as nodes represent computation steps, and edges represent the dependency between computations. In the following work [Hendrickson and Kolda, 2000], the authors surveyed different metrics that can be used as objective functions in partitioning graphs. Their applicability domain was the division of large calculations across processors of a parallel computer. Four graph partitioning models were described: *bipartite graph model*, *hypergraph Model*, *multi-constrain and multi-objective partitioning* and *skewed partitioning*. Reasoning that these four types could more easily be partitioned the authors suggested the use of an existing *multi-level* paradigm for partitioning that consists in generating smaller graphs that preserve the essential properties of the original, devising an algorithm for partitioning the smaller graphs and a refinement technique for improving the partitioning back to the original graph.

For processing large graphs, the authors in [Patwary et al., 2019] suggested a way to partition large graphs that does not involve loading the entire graph into main memory, as that would be computationally expensive and in some cases infeasible. The proposal is a window-based streaming graph partitioning algorithms, WStream. WStream maintains balanced partitions across machines such that the edge-cuts are minimized, in order to save on communication costs between machines. It also makes use of a streaming window for a more informed partition of graph vertices. A similar work to this was proposed, dealing with acyclic graphs instead of generic graphs, where the authors devised an evolutionary algorithm adaptation to provide better candidate solutions incrementally [Moreira et al., 2018].

The work of distributing a graph across different machines, while outside the application of this thesis, provides insight into a generally useful concept of graph partitioning, that is, minimizing edges that are cut and balancing the nodes of the graph across the desired partitions. Effectively, this idea is at the core of the existing graph partitioning algorithm used in EDGE itself [Fontes and

Silva, 2019].

The authors of [Alamdari and Mehrabian, 2012] studied the problem of partitioning a directed acyclic graph by deleting a set of edges with minimal total weight and in a way that the resulting partitions each have just one sink. This problem is known to be NP-hard, and the authors prove that it is even hard to approximate. From this, we understand that we might need to loosen the restrictions on resulting partitions.

Dropping the sink restrictions of the previous problem definition, and proposing new heuristics for refinement and enforcing the acyclic property, a *multi-level* approach was suggested in [Herrmann et al., 2017] for partitioning large graphs. The results from experiments of graphs from an application and from public collection improved about 59% against the state of the art.

Evolutionary dynamics on graphs have been extensible studied from the point of view of population evolution, where each individual is represented by a vertex with connections to other individuals that might be created or destroyed. Aiming at generalizing how population structure affects the evolutionary dynamics, the field of evolutionary graph theory was introduced [Lieberman et al., 2005]. We believe the evolution of graphs in EDGE is an example of such a domain where the topology and fitness of individuals can influence how future generations evolve their own topology.

We present a summary of our findings of other works in graphs in Table 3.3.

Work	PoG	EoG	PoDAG
[Tettamanzi, 1998]	X	X	
[Hendrickson and Kolda, 2000]	X		
[Lieberman et al., 2005]		X	
[Alamdari and Mehrabian, 2012]			X
[Herrmann et al., 2017]			X
[Moreira et al., 2018]		X	X
[Patwary et al., 2019]	X		

Table 3.3: Layout of Graph Theory works on Partition and Evolution of Graphs. PoG (Partition of Graph). EoG (Evolution of Graphs). PoDAG (Partition of Directed Acyclic Graph).

3.4 Similarity Measures

We have given a brief overview of what constitutes a similarity metric in Section 2.7. However, in this section, we take a step towards other works that have made use of similarity metrics, specifically between datasets and search for possible leads on new ways to formulate these types of measurements. The similarity between datasets can be used to process datasets from different sources, plus making use of past successes in similar situations [Oberbreckling and Rivas, 2019].

It is important to refer that such a similarity measure is more challenging when we admit comparisons between datasets that have little to no overlap in their characteristics or content, for example, different number of attributes and their types.

The main research direction focuses on similarity measures being oriented towards the similarity between two data points instead of two whole datasets. The problem of defining a similarity measure between different data points is not trivial, and a number of proposed metrics have been evaluated and compared [Dengsheng Zhang and Guojun Lu, 2003, Hamaker and Boggess, 2004, Kagie et al., 2009]. We take an interest in this problem as well because it can be the source of inspiration and possible adaptations of currently well-established metrics to our needs. The authors of [Dengsheng Zhang and Guojun Lu, 2003] have evaluated the following metrics: *Minkowski-form distance*, *Cosine distance*, χ^2 *Statistics*, *Histogram Intersection*, *Quadratic distance* and *Mahalanobis Distance* for image retrieval. Their results showed that the *Manhattan distance*, which is a special case of the *Mahalanobis Distance* and the χ^2 *Statistics* performed best in determining image similarity. The problem of improving a model by adding distance measures, which is not unlike ours, is discussed in [Hamaker and Boggess, 2004]. The authors extend a classifier using distance measures in order to compare elements in the feature space. The metrics tested were: *Euclidean distance*, *Manhattan distance*, *Overlap*, *Value Difference metric*, *Heterogeneous Euclidean-Overlap metric*, *Heterogeneous Value Difference metric* and *Discretized Value Difference metric*. The experiments were made using several different datasets, and the results showed that the best metrics depended on the dataset. The authors recommended the testing of different metrics for each particular dataset and chose the best for each particular case. Despite this, one of the metrics that performed reasonably well across all datasets was the *Discretized Value Difference metric*. We consider dissimilarity as closely resembling similarity in the sense that from one, we can derive adaptation to the other type of metric. One domain that benefits from quantifying dissimilarity are recommender systems. The authors of [Kagie et al., 2009] experimented with: *Euclidean distance*, *Hamming distance*, *Heterogeneous Euclidean-Overlap metric* and *Adapted Gower Coefficient*. The results showed the *Hamming distance* performing better when recommending only one product and the *Adapted Gower Coefficient* dominating recommendations of three or more products. We can see that one of the most recurring metrics is the Euclidean distance. This metric has proven to be the default in many cases where no further search for metrics is done. One disadvantage of this metric, however, is that it is susceptible to the scale of attributes, first demanding a normalization of the attributes before computing the metric.

Similarity measures for datasets are often prescribed in problems of distributed dated mining, where the data mining task is performed across multiple computational resources and where the clustering of similar datasets can be used to mine specific data from each cluster of similar data [Parthasarathy and Ogihara, 2000]. Dataset similarity was exploited in [Parthasarathy and Ogihara, 2000] using a case study of Census data together with synthetic datasets. The authors proposed a scalable and storage-efficient algorithm that compares datasets based on how their attributes are correlated with each other. The only downside to such an approach is that we assume homogeneous datasets, i.e., that follow that same structure when it comes to the attribute

dimension. An example of such a dataset can be a distributed transactional database.

A patent was filled at the end of 2019 [Oberbreckling and Rivas, 2019] that deals precisely with determining a similarity between different datasets from different sources, taking into account a myriad of perspectives: data similarity, column order, document type, overlapping content, and others. Such a framework encompasses four stages in its primary process component: preparing, enriching, data discovery, and publish. The authors go into specific technological decisions and details, but a fundamental idea that can be seen is the idea of enriching a dataset. From our perspective, this means generating meta-data based on the content of a dataset that would allow datasets with different characteristics to be easily comparable.

Data mining is commonly defined as discovering models for data [Rajaraman and Ullman, 2011]. If we are trying to define a similarity between structurally different datasets, an intuitive process would be to somehow convert both datasets to a common representation such that all the distance and similarity metrics we mentioned before could be applied. Following this idea of converting to a common representation, one thing that is common to any datasets to which models are applied and tested is the evaluation metrics. We might then consider a robust model or set of models and compare how they fare in two different datasets as a way to compare how closely related such datasets are. The foundation for this idea, the statistical comparison of models over different datasets, is presented in [Demšar, 2006]. The authors discuss several manners to compare two classifiers across different datasets, these are: *Averaging over Datasets*, *Paired T-Test*, *Wilcoxon Signed-Ranks test* and *Sign test*. Comparisons of multiple classifiers are also discussed: *ANOVA* and *Friedman test*. Assuming that similar datasets allow for similar results by the same model, we could adapt the work mentioned before to a suite of models, providing the models used are robust in the sense that they are not biased towards dataset-specific patterns.

We present a summary of our findings of other works in similarity measures in Table 3.4.

Work	SMs	DMs	SM-BD	Application of SM
[Parthasarathy and Ogihara, 2000]			X	X
[Dengsheng Zhang and Guojun Lu, 2003]	X			
[Hamaker and Boggess, 2004]	X			X
[Demšar, 2006]			X	X
[Kagie et al., 2009]		X		X
[Oberbreckling and Rivas, 2019]			X	

Table 3.4: Layout of works on Similarity Measures (SMs) and Distance Metrics (DMs). BD (Between Datasets).

3.5 Meta-Learning & Meta-Features

In Section 2.5 we gave a short introduction to the concept of meta-learning. Meta-learning is a research field that encompasses a wide variety of works, and as such, we focus our effort in

describing works that fall more within the scope of this work.

Model-Agnostic Meta-Learning (MAML) is an algorithm that trains the parameters of a model based on gradient descent [Finn et al., 2017]. The model's parameters are trained by sampling batches of tasks, computing the new parameters for each specific task, and then aggregating all directions of the update on the model's initial parameters. With this algorithm, the authors achieved state-of-the-art performance on few-shot classification benchmarks. The only restriction is that the model is trained using gradient descent. Using the knowledge from different tasks and a small number of gradient updates, the algorithm produces models that can quickly learn a new task.

More recently, a meta-learning approach was proposed for ensemble methods. This approach for quickly learning the task of soot density recognition [Gu et al., 2020]. The approach proposed by the authors starts with training a model, in the work presented a convolutional neural network (CNN), using the previously mentioned MAML algorithm on a set of different tasks which are argued as being similar or complements of the main objective task of soot density recognition. The result is general-purpose optimized parameters (GOIP) for the model in question. The second stage of the approach constructs an ensemble based on the GOIP where each individual of the ensemble is further tuned with varying parameter choices. The results showed the taken methodology to improve over the commonly used deep neural networks.

The idea of converting datasets to a common representation is compelling, in the sense that allows the use of standard metrics like the Cosine similarity or the Euclidean distance. A possible approach for a common representation is that of characterizing each dataset by a set of meta-features, such that those meta-features can be compared between datasets to obtain similarity measures. The extraction of meta-features has even been used in order to identify noise in datasets [Garcia et al., 2013].

In [Garcia et al., 2013], several meta-features were used to evaluate how much noise was present in a given dataset. Particularly, measures of complexity were used. Complexity measures try to estimate how difficult a particular classification task is with regard to the available data. The authors experimented with 42 datasets where artificial noise was injected, creating a suite of several datasets with varying degrees of noise. Analyzing histograms with the values of complexity measures for different noise levels and types of noise, two metrics exhibited particularly useful ways to distinguish a dataset's noise level: N1, the fraction of borderline points; and N3, the error rate of the one-nearest neighbor classifier.

In the landmarking domain of meta-features, the authors in [Pfahringer et al., 2000] showed that using these types of meta-features is reasonably effective in placing a learning problem within the place of all possible learning problems, such that the best algorithms for a particular learning task are selected. In their experiments, consisting of both real-world data and artificially created datasets, the usage of landmarking improved on 8 of the 10 meta-learners experimented, when compared with information-based meta-features. The authors also studied the effect of choice in the learners that will serve as landmarks. This was also shown to impact final results and requires further study.

Arguing that the performance of any specific classifier in a dataset is highly dependent on its

parameters, the authors in [Reif et al., 2012] propose an integration of meta-learning concepts to strengthen genetic algorithms in a way that achieves performance similar to a grid search of the space of parameters, with a fraction of the computational cost of grid search. The initialization of the genetic algorithm that optimizes the classifier's parameters is thus chosen to take into account the best parameter combination values of datasets that are considered similar to the one that is being tackled. This approach, much like our proposal, uses a database of existing datasets (their representation) and of information about which models worked well. The results of experimenting in 102 datasets, showed that in most cases, the approach taken leads to statistically significant improvements over the standard genetic algorithms approach (without the use of meta-features).

We present a study summary of other works in meta-learning in Table 3.5. The work presented in [Reif et al., 2012] motivates our bootstrapping based on meta-features proposal.

Work	Meta-Learning	Ensembles	EAs	Meta-Features
[Reif et al., 2012]	X		X	X
[Pfahringer et al., 2000]	X			X
[Finn et al., 2017]	X			X
[Gu et al., 2020]	X	X		
[Garcia et al., 2013]				X

Table 3.5: Layout of works on Meta-Learning. EAs (Evolutionary Algorithms)

3.6 Conclusion

From the review of the literature, we can arguably affirm that, while the areas of evolutionary algorithms and ensemble learning are unquestionably active, there are still gaps for novel ideas and concepts to arise.

One work we consider similar to ours in the sense that evolutionary, ensembles, and graphs are used is the work introduced in [Zhou et al., 2018]. Its key differences are that it evolves just one ensemble, and each model in the ensemble only sees a subset of the data. It also lacks the meta-learning capabilities we endow EDGE. We consider there is a gap in works that combine the topics discussed in Sections 3.1, 3.2, 3.3, and 3.5. To justify this, we compare other work's properties against those of EDGE by the end of this work, in Table 3.6.

Regarding Table 3.6, we can immediately see that our focus was primarily in Evolutionary Algorithms and Ensembles. This is deliberate since we consider EDGE to be deeply rooted in these two areas of research. It is interesting to note that, from what we reviewed, most works that deal with EAs or Ensembles do not intersect with Meta-Learning. Instead, Meta-Learning seems to be a field where the abstraction lies somewhere else. This distinction is evident when one takes into consideration that with Meta-Learning, the attention is more focused on improving the learning process itself. It was also observed that few works cross the previously mentioned vast domains with Graphs Theory.

Work	EAs	Ensemble	Graphs	FM	ML	PD
EDGE	X	X	*		X	X
[Zhou et al., 2018]	X	X	**			X
[Kim et al., 2002]	X	X		X		Using FM
[Wang and Wang, 2006]	X	X				
[Galar et al., 2013]	X	X		X		X
[Ai et al., 2019]	X	X				
[Stanley and Miikkulainen, 2002]	X		X			X
[Lertampaiporn et al., 2012]		X		X		X
[De la Peña Sarracén, 2017]		X				X
[Webb and Zheng, 2004]		X		X		X
[Pfahringer et al., 2000]					X	
[Reif et al., 2012]	X				X	
[Gu et al., 2020]		X			X	

Table 3.6: Comparison of EDGE with similar works on Research Topics of Interest. EA (Evolutionary Algorithm). FM (Feature Manipulation). ML (Meta-Learning). PD (Promoting Diversity). * Between Models. ** Between Models and Data samples.

Despite our review of the literature, just because we fill a gap does not grant any merit to our method. What grants some merit is the fact that the influence of Evolutionary Algorithms gives robustness to finding solutions to our graph ensembles. Ensemble research has established ensembles as the type of model always to experiment without care for the type of problem due to their widely recognizable performance. The idea to encode ensembles as directed graphs and the metaphor to social dynamics gives us diversity and space to innovate while being inspired by real-life phenomena. Of course, this matters little if we lack the mathematical foundation, which is why we define and formulate EDGE and its improvements in a way that can be challenged or altered with rigor. The idea to use similarity measures based on meta-features of datasets to bootstrap the graph ensembles also contributes to using knowledge from the meta-learning domain to enhance our approach. We consider bootstrapping to be in the realm of meta-learning since we intend to bootstrap from different datasets.

Chapter 4

Research Questions & Solution

In this chapter, we start from the research questions mentioned briefly in Chapter 1, extending them and providing insight into what kind of problems arise from each of them. Next, we dive into the solution proposed to tackle the problems and answer the research questions. The solution described herein will focus on a high-level perspective, leaving the concrete implementation to Chapter 5.

4.1 Research Questions

As previously mentioned, we will focus on answering several questions within the scope of this work. From each of these, several problems arise, both in terms of design and in implementation constraints.

Is it possible to evolve the weights between nodes of the graph ensembles, and the weight of each node's prediction in itself, such that the ensembles as a whole become more powerful predictors?

The intuition behind representing the ensembles as weighted graphs allows models to use the predictions of other models in a way that mitigates possible weaknesses in any one model, and strengthens the overall result by weighting the predictions of each node by their respective performance. Evolving just the topology of the graphs, without any updates to the weights, relies too much on the initialization of the weights between nodes to accurately portray the possible trust dynamics in a population. As such, we argue that the weights can be optimized using the same data that was used to train the topology of the graph in order to increase the prediction power of the ensembles.

Various problems arise in answering the aforementioned research question. One problem is finding a reasonable path to update the weights during the evolution process while avoiding problems like overfitting. Overfitting is the problem of performing well on the training data but without

actual generalization capabilities. Another problem is that of deciding which step to take in a generation, whether the topology evolution or the weight evolution, such that both are balanced and, if possible, complement each other.

Can EDGE bootstrap its *Reservoir* with graphs and base models that were used in previous runs, from a different dataset?

The underlying assumption behind this question is that using base models and graph ensembles that worked well in one dataset to bootstrap another, never-before-seen dataset, leads to better results *faster* than without any type of exploitation. Here we use the term exploitation for profiting in performance from the usage of previous runs from other datasets.

If this question can be answered positively, it means that, by having a database of models and graph ensembles performed well on different datasets, we never need to solve a problem (the prediction of a given variable based on a number of features) from scratch, always starting from some knowledge to be exploited. Another interesting take is that this database itself can be ever-growing such that we refine and improve on the bootstrap capabilities themselves.

The problem resides in how to bootstrap EDGE and what differences should there be in exploiting the results of previous runs, compared to the common exploration of the *Reservoir*'s configuration space of base models.

If EDGE can bootstrap itself with a different dataset, what differences are there with respect to the dataset chosen? Is there any way to chose the dataset to use as bootstrap without exhaustive searching all the datasets available?

This question stems from the previous in the sense that testing the bootstrap of EDGE with all available dataset runs in the database becomes intractable and not scalable with the number of existing datasets. Because of this, we try to answer the question of choosing which dataset to use as bootstrap by looking at similarities between the two datasets, the one we are trying to tackle and the one to use as bootstrap, from their meta-features.

The challenge becomes which meta-features to chose. Even in the realm of meta-features, we have distinct groups that can be taken into consideration, from statistical and model-based measures to more general, simple measures [Rivolli et al., 2018]. The ample diversity in meta-features begets a more in-depth analysis of which meta-features might be more suited to our particular problem.

When does EDGE perform well? How is its performance affected with respect to its configuration, or even with respect to the dataset? What insights can be taken from this?

EDGE, like many other prediction models, is susceptible to its own configuration parameters. The question creates the need to understand how its performance varies when some of its parameters are changed. By studying its behavior we posit that some insights can be derived in order to

understand if the bootstrap mechanism is always beneficial and if no, when does bootstrap seem to be unnecessary.

4.2 Solution Proposal

In an effort to find the answer to the research questions posed in Section 4.1, the focus will first be on providing a general solution for EDGE, as well as its architecture. Then we shall specialize in the domains of weight evolution and bootstrapping. Finally, we mention some implementation concerns.

The developments proposed were constructed in a sequential manner, build one on top of the other. Weight evolution is built on top of EDGE; then the bootstrapping is built on top of the weight evolution, taking notice to perform the necessary experiments to support our findings and implementation decisions.

4.2.1 Weight Evolution

The approach taken to solve this problem was to consider weight updating as an optimization problem. The target is to incrementally update the weights between nodes and the weights associated with each node's confidence in itself in a manner that minimizes a loss function. This loss function is calculated, taking into account the error in the probabilities predicted for each of the training sample's target class. In order to mitigate possible overfitting problems, we will perform small updates to the weights at each time.

The weights are formulated as N-dimensional vectors, where N is the number of target classes. In contrast, each node's weight in its own prediction, its *self-confidence*, is a scalar value. With this formulation, we aim to capture situations where specific nodes are particularly good at identifying certain classes. At the same time, the scalar *self-confidence* targets an equilibrium between a node's own prediction and the prediction from its predecessors.

Incorporating the weight updating into EDGE leads to two types of steps during the evolution of the population of ensembles, the topology step, and the weight step. In order to balance these two, the proposed solution is to use a statistical test based on the fitness of the entire population that decides whether to perform the same step as the previous generation or switch from one type to the other. We argue that using both steps in a single generation has the chance to be counterproductive, in the sense that one operation might be destroying the progress of the other. Due to that, we focused on balancing the two operations instead of performing both at the same time.

EDGE's inner operating logic was revisited in order to introduce our decision process for the type of step to be chosen at each generation. The statistical test proposed is a test of the equality of continuous one-dimensional probability distributions, where the probability distributions taken into account are fitness distributions before and after a given step type, in order to decide on the next step. A diagram illustrating EDGE's internal process and the weight updating is showcased in Fig. 4.1.

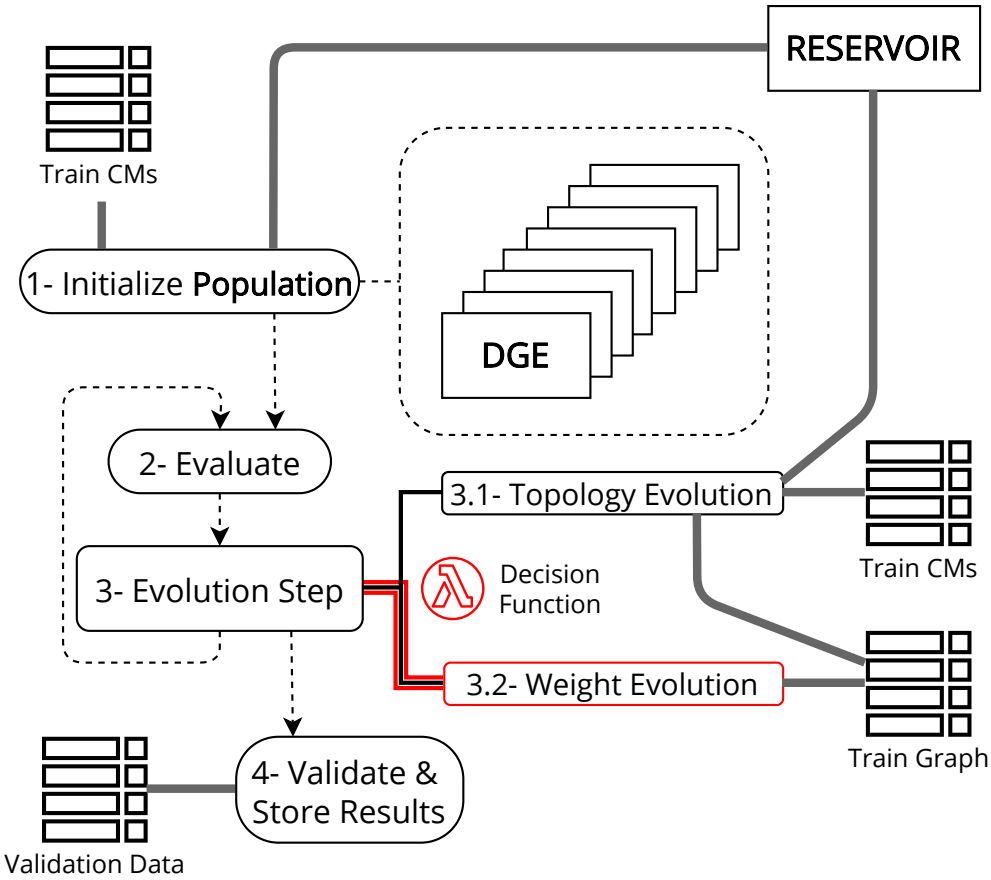

Figure 4.1: Weight updating inside of EDGE, with our contribution outlined.

4.2.2 Dataset Bootstrapping

In order to endow EDGE with bootstrapping capabilities, while leveraging the current architectural design, the solution presented deals with making most of the changes to the *Reservoir* component of EDGE. Instead of the *Reservoir* keeping only a configuration space for the different types of standalone models that can be used as Component Models in the graph ensembles, an auxiliary data structure is kept where several configurations for models, as well as entire graph ensembles, are maintained.

At the end of an arbitrary experiment where EDGE evolved a population of ensembles in a given dataset, a database is filled with the top-performing graph ensembles ordered by fitness and a dataset identification string. The database also keeps a record of a standard representation scheme for the datasets used, for example, an N-dimensional meta-feature vector.

The storage of the best performing ensembles from previous runs allows for bootstrapping capabilities. In contrast, the storage of meta-features for each dataset helps with answering the question of which dataset to chose from in order to increase the effectiveness of the bootstrap, by not having to test all possible datasets as bootstrap.

The exciting thought about this approach is that the database can be incrementally improved by increasing the number of different datasets it has seen, as well as different runs for the same

dataset that might yield better results, while never storing the dataset's entire contents.

A diagram illustrating the new Reservoir is shown in Fig. 4.2. A small example diagram that uses the result from the new features of the Reservoir is presented in Fig. 4.3.

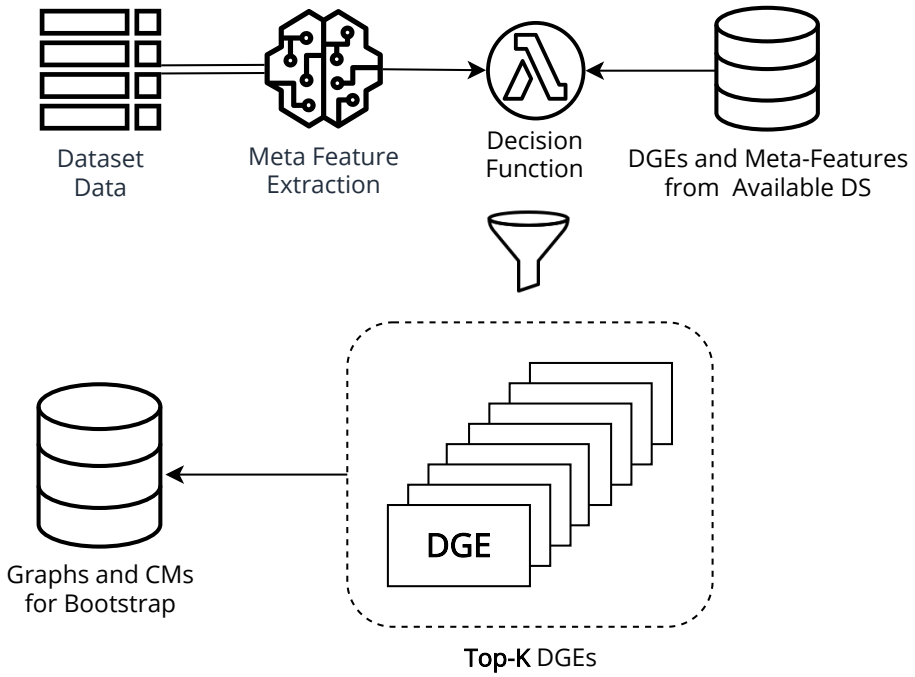

Figure 4.2: Inner working of the Meta Reservoir.

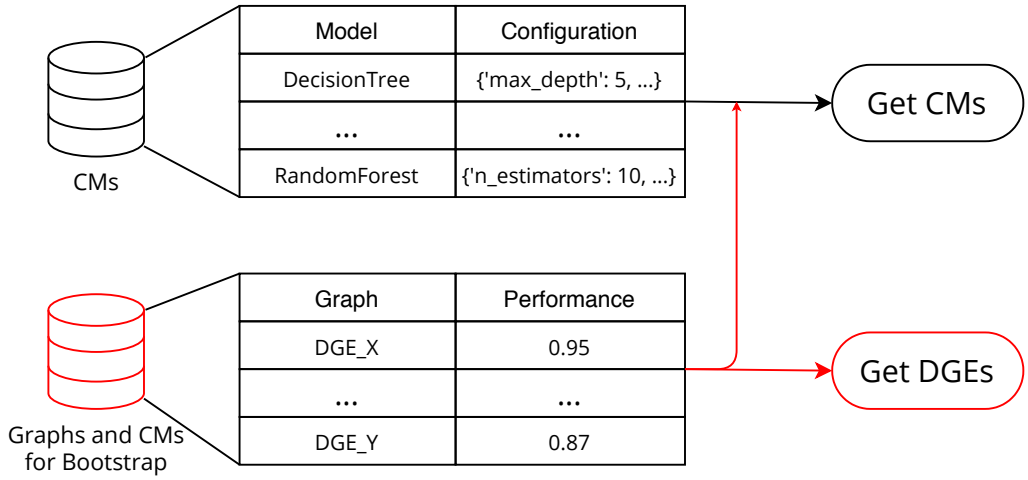

Figure 4.3: Example of the Meta Reservoir, with our contribution outlined.

4.2.3 Implementation Concerns

In order to facilitate all the experiments and the testing of different scenarios, we will focus on making EDGE easy to customize, such that we can quickly deploy different instances of EDGE with different parameters and associated control logic. This will be achieved by encapsulating

most of the core logic in abstract classes while implementing our specific decisions in separate files. We designed checkpoints to be generated every few generations, such that we can analyze EDGE's performance at different stages in time.

To expedite the experiments, we also created several dataset loaders, whose job is to fetch data and convert it into a standardized format, in order to feed EDGE and calculate all the necessary data, whether it be predictions or meta-features.

Chapter 5

Implementation Details

For a complete understanding of the work done, we reserve this chapter to provide a complete description of the EDGE's implementation, together with the trade-offs and compromises that were taken in the implementation of the solution mentioned in Chapter 4. We start by describing EDGE's implementation alongside our improvements to it. Then, we discuss the evolution of the weights and the bootstrapping with different datasets.

5.1 Core Implementation Concerns & Improvements

EDGE is implemented in Python [Van Rossum and Drake, 2009]. In order to fulfill the solutions mentioned in Chapter 4, we rebuilt parts of the core codebase of EDGE, taking care to make everything more modular and as future-proof as possible.

There are three main classes that constitute EDGE's structure, *EDGE*, *DGE* and *Reservoir*. *EDGE* is the one that runs the whole evolution process; it is also the one that is called with all the necessary parameters to initialize the evolution. Some of its key responsibilities include: initializing the population, managing the population through evolution, evaluating the graph ensembles, and saving the final results. In our work, we felt it was necessary to implement some desired features for present and future use. We added the ability to save checkpoint results, allowing the analysis of EDGE's performance over time, and the storage of the best performing models in a database. The database will be further explored in Section 5.3.

Each individual of the population represented as a *DGE* instance is responsible for computing the prediction of a given input X, taking into account its graph structure and the weights associated with it. It previously had the responsibility of deciding how to change its own topology, which we deemed unnecessarily convoluted. As such, we encapsulated the entire logic for the topology evolution in its own class in a way that is abstracted from the *DGE* and, at the same time, allows for other parties to derive and extend the topology evolution logic more easily. A responsibility we added to the *DGE* was that of storing its graph weights, as well as each node's self-confidence,

in a way that allows subsequent gradient calculation with respect to these values, for our weight evolution proposal. The evolution of weights will also be further explained in Section 5.2.

The *Reservoir* is in charge of supplying the base models necessary to create the *DGE* instances, more specifically, the nodes in the graph ensembles. It initially kept a list of all possible combinations of base models and picked at random whenever it was called by *EDGE*. Our idea was to improve upon it by abstracting its behavior completely, such that it can be implemented in any way that is required, while fully working with EDGE. We adapted its initial implementation into a derivation of the introduced abstract class and put forth our implementation where the *Reservoir* supplies not only base models, but complete graphs as well, from a database of previous runs. This last point will be clarified in Section 5.3.

As for other features that we judge to be of relevance to present, we take special notice of the attention to modularity and the focus on easy experimentation. For the modularity, as it was previously alluded to, we focused on encapsulating most of the logic in abstract classes and implementing EDGE's core code together with our developments deriving from these abstract classes. The result is a system where any one piece can be extended, and the system as a whole works just the same, avoiding the need to change anything more than desired. In favor of modularity and settling some technical debt towards the project, we also changed the file hierarchy to a more intuitive way, taking inspiration in libraries like Scikit-Learn [Pedregosa et al., 2011]. Developments to ease experimentation were also introduced. We created a general parser that works on upwards of 150 KEEL datasets [Alcalá-Fdez et al., 2011], in order to efficiently feed many different classification datasets to EDGE.

As intended, the entirety of EDGE will be made publicly available [1] to foster further developments and novel adaptations.

5.2 Weight Evolution

The graphs are still being manipulated using a known graph library, networkX [Hagberg et al., 2008], for its ease of use and fast implementation. The key difference now is that the weights of the graphs and the self-confidence parameter of each node in the graph are being stored as Tensors, from pytorch [Paszke et al., 2019], an open-source machine learning library. The weights are N-dimensional tensors, where N is the number of target classes such that the prediction of each node is element-wise multiplied by the weights associated.

As for the evolution of weights, we used the Adam [Kingma and Ba, 2015] optimization algorithm to evolve the connections between nodes of each graph. The parameters updated using the optimization algorithm were the weights of the edges of each node in its predecessors, along with the self-confidence values. As such, we try to further evolve the graph as a whole by changing the trust each node has in its predecessors instead of using fixed values. The loss is computed as the Mean Squared Error for each class probabilities, aggregated by summing across all the class probabilities.

[1] https://github.com/xfontes42/EDGE

The decision process for the evolution step type was made using the Kolmogorov–Smirnov test [Conover, 1999] (K-S test) to decide whether to switch the type of evolution step or not. At each generation after the first, we use the K-S two-sample test to ascertain whether two distributions of fitness values, the current generation's and the previous', differ from one another. The devised heuristic always starts with a topology step.

In Chapter 6, we go more into detail about the values chosen for the algorithms.

5.3 Dataset Bootstrapping

The Reservoir changes were implemented by deriving from the previously established Reservoir abstract class and changing its inner working to communicate with a SQLite database. The connection to the database is also abstracted by implementing an abstract class that defines the needed methods to fetch the necessary data. The reason for this is that other interested parties might want to use different database systems to store the graphs at the end of evolution. The SQLite database was chosen to facilitate rapid prototyping and because, at the time of this writing, EDGE is being used and tested in purely academic settings.

The SQL schema was designed to be extremely simple to understand, seeing that we store only a table that associates the dataset to its meta-feature representation, a table that associates the dataset id with the fitness values and the respective graph ensemble, and a table that keeps track of the base models. The utilized schema is shown in Figure 5.1.

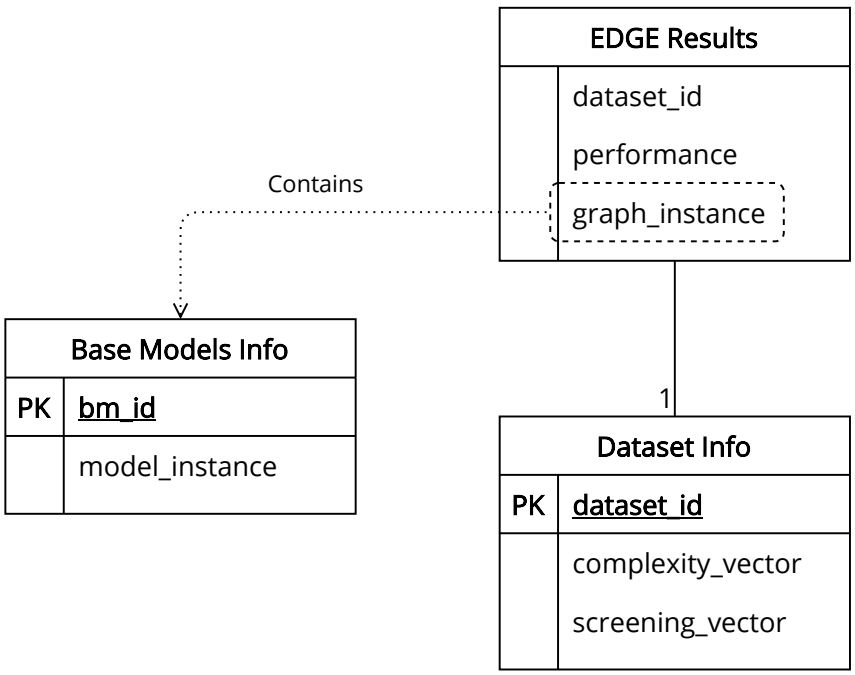

Figure 5.1: Database schema for Bootstrapping.

The meta-feature extraction is done by using a python library pymfe [Rivolli et al., 2018] that allows the automatic extraction of several meta-features, with a wide array of options at our

disposal. We consider the similarity of two datasets as a function based on meta-feature values for the two datasets, which is to be minimized. In Chapter 6, we go more into detail about the values that parameterize the Reservoir and our choice of meta-features.

Chapter 6

Experimental Results

This work will be evaluated with respect to our two main developments, the evolution of weights of the graph ensembles in EDGE, and the study of the usage of meta-features to propose a similarity measure between datasets to provide EDGE with bootstrapping of base models and graphs. A benchmark analysis will also be designed in order to compare EDGE with baseline models.

6.1 Core Setup

In this section, we will describe the set of experiments designed and each question that the experiments aim to answer, together with the basic configuration of EDGE that was used. EDGE's configuration can be summarized in two tables. Table 6.1 showcases the parameters of EDGE itself, with two of them, the *PopSize* and *Exploit Value*, varying with different possibilities. The first five parameters were chosen to take into consideration the values presented when EDGE was first introduced [Fontes and Silva, 2019]. Here we vary the population size because we believe it is worth studying how does the size of the population of ensembles affect the final performance, considering the size of the population directly impacts the computational cost of running EDGE. The next three values are the ones that were chosen to deal with the weight evolution, a small learning rate and a small number of optimizer steps were chosen in an attempt to mitigate possible overfitting issues. The K-S value comes from preliminary testing on a random dataset. While we admit that these parameters were not fully explored, we argue that our choice is a reasonable one within this context. The subsequent parameter, *Exploit Value*, was also varied between 0%, 50% and 100%. The 0% is merely the ablation of the bootstrap mechanism, while the 100% entails the complete reliance on the models and graphs from bootstrap. The 50% is the middle ground, where we aim to explore the Reservoir's configurations, displayed in Table 6.2, while also exploiting the results from bootstrapping. Finally, the last three variables represent how we divided our data. A conventional split for the train-test is 80-20, so we maintained the 20% as data never to be trained on by EDGE and to calculate our performance metrics. The remaining 80% were divided such that

we reach an acceptable compromise between training the base models, and training the graphs' topology and weights. We chose the compromise to be 60% for the base models and 20% for the graphs' evolution (with the remaining 20% for validation).

Table 6.1: Configuration space of EDGE.

Parameter	Values
NElite	1
NBottom	0
MutationRate	0.2
PopSize	10, 20, 40
NGenerations	50
Adam Learning Rate	0.001
Optimizer Steps	2
K-S p-value	0.5
Exploit Value (%)	0, 50, 100
Train CMs examples (%)	60%
Train DGEs examples (%)	20%
Test examples (%)	20%

Table 6.2: Configuration space of EDGE's Reservoir of models.

Parameter	Gradient Boosting	Random Forest	Decision Tree
Training examples (%)	60	60	60
Estimators	5, 10, 15, 20	5, 10, 20, 30	-
Criterion	MSE Friedman MSE	Info. Gain Gini Impurity	Info. Gain Gini Impurity
Max Depth	5, 10, 15	5, 10, 15	5, 10, 15, 20, 50
Max Features	All Features	All Features Sqrt	All Features Sqrt, Log2 0.33, 0.5, 0.6
Learning Rate	0.001, 0.01, 0.1	-	-
Loss	Deviance	-	-
Splitter	-	-	Best, Random
Min. Samples for Split	-	-	2, 4, 8, 16

A baseline suite of models was designed and tested on the same datasets as EDGE in order to understand where does EDGE fare better than the baseline and if there are conclusions that can be drawn based on the features of the datasets themselves. The configuration for the baseline is shown in Table 6.3. The configuration space is deliberately similar to that of EDGE's Reservoir in order to serve as an adequate comparison term. One of the key differences is the number of examples trained on by the baseline. Since we set EDGE to use 60% of the data to train the base models and 20% to train the graph ensembles, we argue that the baseline should take into account the same total percentage of data, 80%. Both the parameter of the max number of features and the

minimum amount of samples for a split are reduced when compared to EDGE's Reservoir, but this was merely a design decision since we argue that the extra combinations would not be as relevant to test. When presenting and discussing the results, the baseline was grouped based on the type of model (DT, RF, or GB) for a better interpretation of which type of baseline models performs better in each dataset.

Table 6.3: Configuration space of baseline test suite. In **bold** are the differences to the configuration space of EDGE's Reservoir.

Parameter	Gradient Boosting	Random Forest	Decision Tree
Training examples (%)	**80**	**80**	**80**
Estimators	5, 10, 15, 20	5, 10, 20, 30	-
Criterion	MSE Friedman MSE	Info. Gain Gini Impurity	Info. Gain Gini Impurity
Max Depth	5, 10, 15	5, 10, 15	5, 10, 20, 50
Max Features	All Features	All Features Sqrt	**All Features** **Sqrt, Log2**
Learning Rate	0.001, 0.01, 0.1	-	-
Loss	Deviance	-	-
Splitter	-	-	Best, Random
Min. Samples for Split	-	-	**2**

The majority of the datasets used come from the KEEL Dataset Repository [Alcalá-Fdez et al., 2011]. Another 4 datasets were used in one of the experiments, the same datasets used by the authors when EDGE was first introduced [Fontes and Silva, 2019]. Due to the continuing nature of this work, different sections of the experiments made use of different subsets of the collection of datasets presented. We make that note on the table that summarizes all the datasets, using an identification name and some general information about each dataset's features, in Table 6.4.

We will discuss this choice for datasets within the respective experiments. Nonetheless, the datasets chosen reflect both the scope of this work and resource constraints. All the results showcased are the corresponding average of 5 runs for the baseline, and 3 runs for EDGE. MNIST contains 100k samples that are usually divided into 70k for training and 30k for testing. Because we used a library to facilitate the downloading of data, what we believed to be the full MNIST dataset, was just the part typically used for training, hence the 70k samples in Table 6.4. We claim this is not an issue since we are using only MNIST to compare against the initial implementation of EDGE, which also used the same 70k samples.

The naming scheme for the KEEL datasets has been made such that the prefix denotes from which section of the KEEL each dataset is, where: *ST* is from the standard portion, *L9* is from the lower than 9 imbalance ration part and *H9* from the higher than 9 imbalance ratio. The reason for this choice was just to increase the number of datasets that are usable to us since we had to implement a KEEL dataset parser from scratch that read the raw data files provided in the KEEL

Table 6.4: Summary of datasets used. Marked with * are the datasets used in Section 6.3, Section 6.4 and Section 6.5.

Dataset Name	No. Samples	No. Features	No. Target Classes
MNIST	70000	784	10
Anuran	7195	22	10
Appliances	19735	30	10
Parking Lot	20448	1067	10
ST-tae	151	5	3
ST-zoo	101	16	7
ST-hepatitis	80	19	2
ST-appendicitis	106	7	2
ST-newthyroid	215	5	3
ST-iris	150	4	3
ST-monk-2	432	6	2
ST-wine	178	13	3
ST-shuttle-c2-vs-c4	129	9	2
ST-ring	7400	20	2
L9-iris0	150	4	2
L9-newthyroid2	215	5	2
L9-new-thyroid1	215	5	2
H9-zoo-3	101	16	2
H9-winequality-white-9_vs_4	168	11	2
H9-poker-8-9_vs_5	2075	10	2
H9-car-good	1728	6	2
ST-titanic *	2201	3	2
ST-haberman *	306	3	2
L9-haberman *	306	3	2
ST-saheart *	462	9	2
ST-led7digit *	500	7	10
ST-balance *	625	4	3
L9-pima *	768	8	2
ST-mammographic *	830	5	2
ST-vehicle *	846	18	4
ST-german *	1000	20	2
ST-flare *	1066	11	6
H9-flare-F *	1066	11	2
ST-contraceptive *	1473	9	3
ST-yeast *	1484	8	10
L9-yeast1 *	1484	8	2
H9-abalone19 *	4174	8	2
ST-yeast-1-4-5-8_vs_7 *	693	8	2
H9-winequality-red-4 *	1599	11	2
H9-abalone-19_vs_10-11-12-13 *	1622	8	2
ST-marketing *	6876	13	9
ST-cleveland *	297	13	5

website [1].

[1] http://www.keel.es/

With regards to the experiments that have used bootstrap, when we mention previous runs, we are referring to experiments that were made without any bootstrap and with a smaller total number of generations. Each run stored the top-5 graph ensembles and their respective CMs in the database.

6.2 Experimenting with Weights

In this section, we compare the usage of our proposal for weight evolution against known results for EDGE in the 4 datasets presented by the authors, and against the baseline suite of models. We alter slightly the parameters shown in Table 6.1, to better compare against the results presented in [Fontes and Silva, 2019]. In order to better distinguish between the implementations of EDGE, we are naming our implementation *in this section* as Weighted-EDGE. The parameters of each implementation are shown in Table 6.5. It is worth noting that in this entire section Weighted-EDGE uses no bootstrapping capabilities, in order to test the individual contribution of the evolution of weights.

Table 6.5: Parameter values for Weighted-EDGE and EDGE. NA (Not Applicable).

Parameter	Weighted-EDGE	EDGE
NElite	2	2
NBottom	0	0
MutationRate	0.1	0.1
PopSize	20	20
NGenerations	100	100
Adam Learning Rate	0.001	NA
Optimizer Steps	2	NA
K-S p-value	0.5	NA

The 4 datasets referred previously are: MNIST [Deng, 2012], a handwritten digit classification dataset; Anuran [Colonna et al., 2017] dataset, of classification of anuran species based on characteristics of their callings; Appliances [Candanedo et al., 2017] data, a 10-class discretized time series dataset; and Parking Lot [2], a synthetic dataset generated for the occupancy of a student parking lot. The results for these 4 datasets can be shown in Table 6.6.

The results from the KEEL datasets, where Weighted-EDGE is compared with the baseline model suite, is shown in Table 6.7. We opted for grouping the baseline models by their model type, just for a complete picture of where and when Weighted-EDGE outperforms the baseline.

Discussion

Starting with the comparison between Weighted-EDGE and EDGE, we see that Weighted-EDGE manages to improve on 3 of the 4 datasets, by an average margin of 4.20 percentage points. The

[2]Dataset available online from https://github.com/xfontes42/parking_lot_ds

Table 6.6: EDGE with and without weight evolution comparison. Results from EDGE are the best results presented in [Fontes and Silva, 2019]. Between parentheses (*value*) we put the best result of all runs from Weighted-EDGE.

	EDGE		Weighted-EDGE	
Dataset	Accuracy	F1-Score	Accuracy	F1-Score
MNIST	97.78	97.78	**97.93** (98.34)	**97.93** (98.34)
Anuran	**99.17**	**99.16**	98.80 (98.94)	98.79 (98.94)
Appliances	86.27	84.29	**93.37** (93.96)	**93.26** (93.74)
Parking Lot	87.68	87.10	**93.28** (93.81)	**93.24** (93.75)

fourth dataset, where it underperforms, lags only by about 0.4 percentage points. Taking a closer look at the three datasets where it performed better than EDGE, we see that one of those was just by an average of 0.20 percentage points (and 0.56 percentage points in the best case). In comparison, in the two datasets that represent regression tasks that have been discretized, its gains are significantly higher, with an average gain on the two datasets of 6.35 percentage points. Nonetheless, we consider the weight evolution to be a success since it competed or improved on the accuracy performance of the dataset. We also mention the results for the F1-Score, which present similar situations as the accuracy values already mentioned.

The results from the comparison between Weighted-EDGE and the baseline are encouraging, with EDGE achieving the best result in 34 out of the 38 datasets used, with gains as large as 30 percentage points. The datasets where EDGE was beaten did so by an average margin of 0.52 percentage points, which we believe is not cause for concern because we are comparing results where all accuracy values are over the 98% mark. There seems to be a trend where the worse the performance of the baseline is, the better are the gains when checking the results of EDGE. This is not entirely unsurprising since we can intuit that we have much more room to improve from 30% or 60% in accuracy than what we have when the baseline already achieves results on the order of upwards of 85% in accuracy.

There are some fringe cases that we want to address as well. These are datasets where the best accuracy achieved was 100%. Achieving 100% accuracy, while interesting, might be because we have too small or too easy to classify datasets. When we take a look at the datasets where such good results were achieved, we notice that all of them, except for one, have less than 250 data points.

Overall, the results seem to support the answer that indeed, weight updating is a strategy that, when incorporated with EDGE, leads to extremely powerful ensembles, even when compared to virtually the same models that EDGE as used in its Reservoir. It would also be fruitful to run the comparison made with the initial four datasets across the other 38. However, such a comparison would be too time-consuming for the scope of this work. Henceforth, we will assume Weighted-EDGE is the *de facto* implementation referred to in the rest of the document, and unless specified, as EDGE.

Table 6.7: Baseline versus Weighted-EDGE, in terms of Accuracy. The best results are **bolded**.

Dataset Name	Baseline DT	Baseline RF	Baseline GB	Weighted-EDGE
ST-tae	69.03	63.87	62.58	**84.65**
ST-zoo	**98.10**	97.14	95.24	98.04
ST-hepatitis	91.25	88.75	87.50	**98.33**
ST-appendicitis	88.18	90.91	81.82	**94.34**
ST-newthyroid	97.21	**99.07**	97.67	98.77
ST-iris	96.67	96.00	96.67	**100.00**
ST-monk-2	**100.00**	**100.00**	**100.00**	**100.00**
ST-wine	96.11	98.33	94.44	**99.63**
ST-shuttle-c2-vs-c4	**100.00**	**100.00**	**100.00**	**100.00**
ST-ring	90.50	95.27	94.59	**97.35**
L9-iris0	**100.00**	**100.00**	**100.00**	**100.00**
L9-newthyroid2	**100.00**	**100.00**	99.53	**100.00**
L9-new-thyroid1	98.60	98.14	97.67	**99.38**
H9-zoo-3	**100.00**	98.10	95.24	98.69
H9-winequality-white-9_vs_4	**100.00**	**100.00**	**100.00**	99.60
H9-poker-8-9_vs_5	98.94	98.89	98.80	**99.42**
H9-car-good	99.19	99.19	99.42	**99.61**
ST-titanic	77.78	78.32	79.14	**79.29**
ST-haberman	73.87	71.61	74.19	**89.11**
L9-haberman	79.03	74.84	74.19	**88.02**
ST-saheart	70.97	69.46	65.59	**87.16**
ST-led7digit	74.20	75.40	74.40	**76.67**
ST-balance	81.92	86.72	86.56	**93.72**
L9-pima	72.99	74.55	77.27	**91.58**
ST-mammographic	86.75	86.75	86.14	**89.24**
ST-vehicle	73.65	78.00	74.35	**90.23**
ST-german	74.10	77.00	76.40	**90.07**
ST-flare	72.90	73.55	73.08	**81.93**
H9-flare-F	95.89	95.98	95.79	**97.06**
ST-contraceptive	54.58	55.86	54.31	**78.61**
ST-yeast	58.52	63.43	60.34	**83.38**
L9-yeast1	77.78	80.47	78.72	**89.49**
H9-abalone19	99.28	99.28	99.28	**99.74**
ST-yeast-1-4-5-8_vs_7	96.40	95.83	96.40	**98.27**
H9-winequality-red-4	96.56	96.56	96.56	**98.79**
H9-abalone-19_vs_10-11-12-13	98.15	98.15	98.15	**99.18**
ST-marketing	36.12	36.10	35.87	**67.70**
ST-cleveland	61.67	62.00	55.00	**82.55**

6.3 Experimenting with Bootstrap

In this section, we take for granted the evolution of weights and test the approach of bootstrapping EDGE with DGEs and CMs that were the final result of runs using other datasets. Due to resource constraints, and because we tested the bootstrapping of one dataset with every other dataset, we

settled on a subset of the datasets to test on, highlighted in Table 6.4. The baseline is compared with EDGE, and the best results are shown in Table 6.8.

Table 6.8: Baseline versus EDGE with several degrees of Bootstrap, in terms of Accuracy. The best results are **bolded**. Between parentheses $(X-Y)$, we denote the generation where it improved over the Exploit 0%, as X, and the generation where it first achieved it's highest value, as Y. The values between parentheses only appear on situations where the Accuracy is bigger than the baseline.

Dataset Name	Baseline	Exploit 0%	Exploit 50%	Exploit 100%
ST-haberman	74.19	**89.11**	88.45	89.05
L9-haberman	79.03	88.02	89.11 (5-30)	**89.54** (5-40)
ST-saheart	70.97	87.16	**88.89** (5-30)	88.74 (5-50)
ST-led7digit	75.40	76.67	76.53	**78.70** (5-40)
ST-balance	86.72	93.72	93.61	**94.49** (5-20)
L9-pima	77.27	91.58	**92.27** (10-20)	91.15
ST-mammographic	86.75	89.24	90.04 (5-50)	**90.30** (5-20)
ST-vehicle	78.00	90.23	**91.84** (5-35)	91.13 (5-50)
ST-german	77.00	90.07	**90.73** (5-40)	90.35 (5-10)
ST-flare	73.55	81.93	**83.30** (5-35)	82.36 (5-35)
H9-flare-F	95.98	97.06	97.25 (5-5)	**97.75** (5-50)
ST-contraceptive	55.86	78.61	**80.55** (5-50)	79.34 (5-50)
ST-yeast	63.43	83.38	**83.65** (15-50)	83.25
L9-yeast1	80.47	89.49	90.79 (5-15)	**91.07** (5-50)
ST-titanic	79.14	**79.29**	78.84	78.54
H9-abalone19	99.28	99.74	99.66	**99.75** (5-5)
ST-yeast-1-4-5-8_vs_7	96.40	98.27	**98.56** (5-20)	98.41 (5-50)
H9-winequality-red-4	96.56	98.79	**98.81** (5-5)	98.60
H9-abalone-19_vs_10-11-12-13	98.15	99.18	**99.38** (5-5)	99.26 (5-50)
ST-marketing	36.12	67.70	67.50	**67.89** (5-50)
ST-cleveland	62.00	82.55	**86.35** (5-15)	85.40 (5-25)

Discussion

From the analysis of Table 6.8 we can see that endowing EDGE with bootstrapping capabilities seems to have been successful in most cases. On 19 out of the 21 datasets tested, EDGE with any type of exploitation improved an average of 1.08 percentage point, while lagging in the remaining 2 datasets by an average of 0.26 percentage points. The absolute values of the improvements were not as significant as in Section 6.2, but we were not expecting them to be since the margin for improvement at this stage is even smaller than when taking a look at the baseline. Nonetheless, the conclusion from a first analysis is that bootstrapping further improves the predictive skill of EDGE as a whole.

Inspecting the differences between the results of relying only on bootstrap (*Exploit 100%*) and taking a balanced approach between exploitation and exploration (*Exploit 50%*), we find that using

the latter approach led to the best result in 11 of the 19 wins, while the remaining 8 going to the former. We try to explain this difference with the hypothesis that, while exploiting previous data is beneficial, some degree of exploration is needed so that we can outgrow of possible local optimum values.

When also considering the two extra values added on Table 6.8 whenever using bootstrapped improved the prediction accuracy, we see that it quickly surpassed the situations with Exploit 0%. Generation 5 is the first checkpoint that is taken during the course of the evolution, and it is at this generation number that most of the time, using exploit surpassed the Exploit 0%, while only reaching their highest value (the second number between parenthesis) much later.

In these experiments, each dataset was individually bootstrapped with all the other datasets, which amounts to $N * (N-1)$ different experiments (each run 3 times and averaged). The $N-1$ term is because we never bootstrap a dataset with previous runs from itself since that could be akin to using graphs and base models which are already known to perform well on the given data. This quadratic cost of computation is not further explored here. However, in later sections, we will propose a method for choosing which dataset to bootstrap with, without needing to exhaustively search all possible combinations.

Taking everything into account, we reason that using any degree of bootstrap is better than not using any. Not only is it better in terms of values achieved but also in how quickly it surpasses the situations where bootstrap was not used. It is also worth mentioning that, as the database evolves in terms of datasets, the results of bootstrapping should also increase, since we have more prior knowledge that can be exploited.

6.4 The impact of EDGE's parameters on its performance

The previously presented results all show the best results for each dataset, compounding all the configurations of EDGE that were tested during the experiments. In this section, our focus is not so much to see if EDGE is better than the baseline, but *when* is EDGE better than the baseline. What are the effects of its parameters, namely the *Exploit Value* and the *Population Size*. For this, we averaged the best results of all the datasets that are marked with *****, grouping by *Exploit Value* and *Population Size*. The results are summarized in Table 6.9.

Table 6.9: Median accuracy of EDGE on several datasets, grouping by parameters *Exploit Value* and *Population Size*.

	PopSize = 10	*PopSize = 20*	*PopSize = 40*
Exploit = 0%	88.67	88.92	89.24
Exploit = 50%	89.16	90.04	89.80
Exploit = 100%	89.60	89.95	90.30
Baseline		78.00	

Discussion

Reviewing Table 6.9 we can see two different patterns emerging. For one, the median accuracy increases with population size. As for the other, the same situation seems to occur when the exploitation degree increases. The only exception to these two patterns is the case for the population size of 20 and the exploitation value of 50%. We believe this exception happens because when we are using exploitation balanced with exploration, there is not such a need for large populations.

It should be noted that these are median values taken across all 21 datasets that are highlighted in Table 6.4, and as such, variation can and will probably exist depending on which dataset is being considered. Other parameters might also influence the performance of EDGE, but we reserved our attention for parameters that more drastically affect the working of EDGE in terms of resource usage (*PopSize*) and inner working (*Exploit*).

In short, and as was the case for the previous section, we see that using any kind of exploitation leads to more powerful graph ensembles than not using any. At the same time, we believe that increasing the population size can be useful in producing stronger ensembles, at the cost of computational resources. Putting the two parameters together, we can trade some performance by using smaller populations, using exploitation to lessen the performance lost.

6.5 Analysing the Bootstrap and Meta-Features

We now focus our attention on the case where all the graphs and base models obtained from the Reservoir come from the bootstrapping dataset, i.e., where *Exploit Value = 100%*. We also fix the population size to be 40, because, from Table 6.9, its the value that has higher gains when compared against using no bootstrap, 1.06 percentage points versus 1.03 and 0.93 for population sizes of 20 and 10, respectively.

The goal with this analysis is to understand if any of the gathered meta-features from the datasets could be used to decide which dataset to use as bootstrap. The experiments were made such that, for the subset of datasets mentioned, each dataset has been bootstrapped with each other dataset, allowing for a more in-depth exploration of the obtained results.

The first step in our analysis was to visualize how well each dataset performed when using each of the other datasets as bootstrap. The results are shown in Fig. 6.1, where we can distinctively see that some datasets benefit from bootstrap, as is the example of dataset 3 being bootstrapped with dataset 5 or dataset 20 being bootstrapped with dataset 3. Some datasets also appear to not benefit at all from bootstrapping, as is the case with dataset 15.

If we disregard the magnitude of the improvement and focus only on relative improvement for each dataset, by normalizing each row between 0 and 1, we arrive at Fig. 6.2. Within this figure, we can more clearly see which datasets served best as bootstrap. The case of dataset 15, which seemed to not benefit at all from bootstrap in absolute terms (from analysis of Fig. 6.1), seems to benefit, even if only by a small margin, of using bootstrap by dataset 8, dataset 11, and dataset 20.

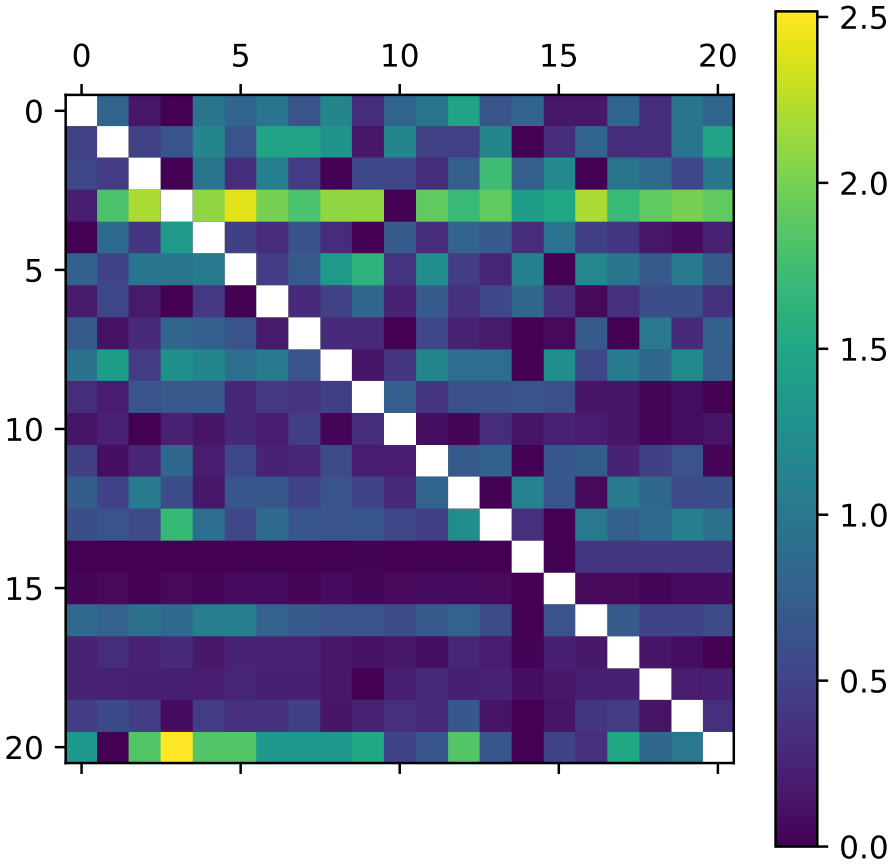

Figure 6.1: Bootstrapping each dataset with each other. Each row represents a dataset and each column the dataset used as bootstrap. Gained Accuracy in percentage points, grouped by each dataset.

Considering the presented analysis, we take further steps in the case regarding meta-features. The meta-features extracted are from two domains, landmark, and complexity. These two choices stem from two different ideas that we believe might complement each other. The landmark meta-features are based on efficient and quick models. Our assumption is that similar datasets might have similar performances on these models, and thus the models can serve as quick screening tests. This assumption is similar to the notion of case-based reasoning (CBR) [Kolodner, 1993]. On the opposite side, the measures from the complexity domain were chosen in order to represent datasets by estimating how difficult it is to separate its data points in its classes. A short list of the meta-features that were chosen is presented in Table 6.10, with a small description for each meta-feature.

In order to understand which, if any, meta-features have the potential to be used to facilitate the

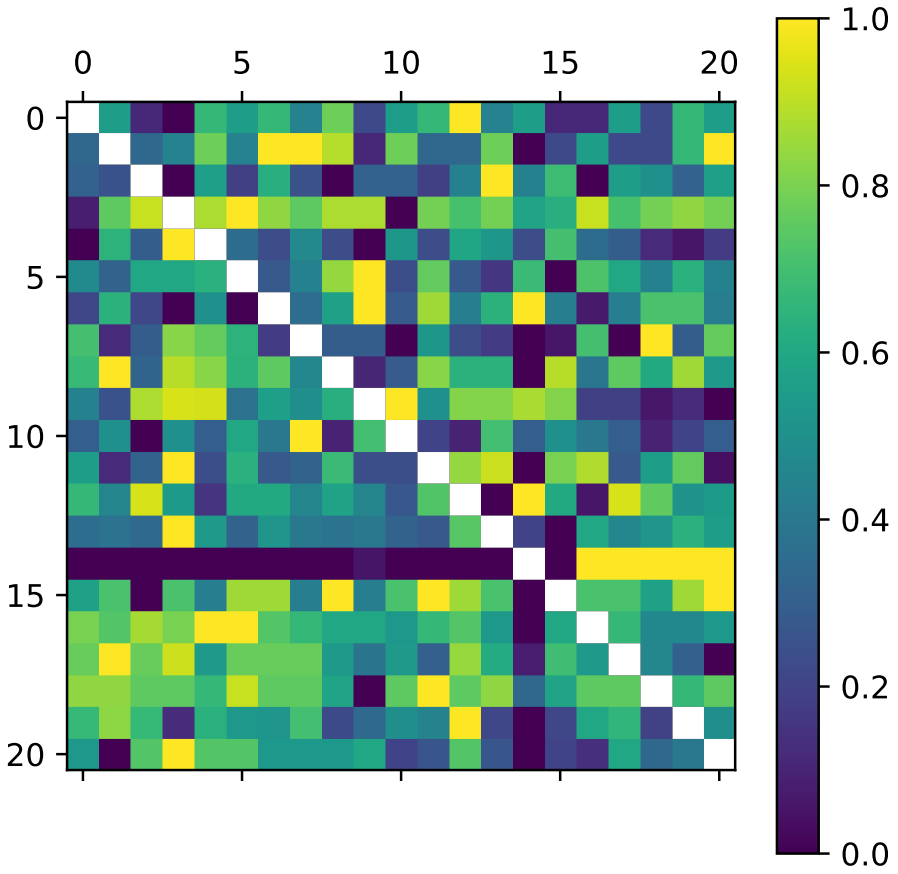

Figure 6.2: Bootstrapping each dataset with each other. Each row represents a dataset and each column the dataset used as bootstrap. Each row's gains in accuracy are normalized between 0 and 1.

choice of bootstrapping, we took all the experiments and associated each with the corresponding distance between the two datasets in question. The rationale behind this analysis is that we are looking for negative correlations of the differences between datasets metrics, such that, for a given metric M_x, for two arbitrary datasets A and B, the smaller the value of $|M_x(A) - M_x(B)|$, the higher the gains of using dataset B to bootstrap dataset A. The resulting analysis is a correlation between each absolute difference of meta-feature values and the gains associated, presented in Table 6.11. The gains mentioned are the values of Fig. 6.2.

Taking a closer look at Table 6.11, see that none of the meta-features exhibited the type of behavior we were expecting, at least not as strongly as hypothesized. The two that showed the negative correlation we were expecting with more strength were *linear_disc* and *F3*. On the other side, two meta-features appeared to correlate positively with the normalized gains, *C1* and *C2*. With

Table 6.10: List of meta-features chosen, aggregated by their corresponding type. DT (Decision Tree). 1-NN (1-Nearest Neighbor). SVM (Support Vector Machine). PCA (Principal Component Analysis). For a more in-depth look at each meta-feature, we refer the reader to [Rivolli et al., 2018, Lorena et al., 2019].

Domain	Meta-Feature	Description
Landmark [Rivolli et al., 2018]	best_node	DT, with most informative attribute
	elite_nn	1-NN, with subset of most informative attributes
	linear_discr	Linear Discriminant, with all attributes
	naive_bayes	Naive Bayes, with all attributes
	one_nn	1-NN, with all the attributes
	random_node	DT, with a random attribute
	worst_node	DT, with least informative attribute
Complexity [Lorena et al., 2019]	C1	Entropy of class proportions
	C2	Imbalance Ratio
	F3	Maximum Individual Feature Efficiency
	L2	Error Rate of Linear Classifier (SVM)
	N1	Fraction of Borderline Points
	T2	Average number of features per dimension
	T3	Average number of PCA dimensions per points
	T4	Ratio of the PCA dimenstion to original dimension

these last two, we believe that something like maximizing, instead of minimizing, the absolute difference between meta-features of two datasets, can effectively serve as a bootstrap measure. As such, we move forward with the next step in our analysis, focusing on the four highlighted meta-features of Table 6.11: *linear_disc*, *F3*, *C1* and *C2*.

We now propose four different preliminary similarity measures, each of which based on one of the meta-features chosen previously. The similarity measures are proposed, such that smaller values indicate a high degree of similarity (as such, will be prioritized the choice for dataset to bootstrap). The proposed measures are presented in Table 6.12

Finally, we explore the results of bootstrap, had our four similarity metrics been used to decide which dataset to bootstrap with, selecting the options that minimize the proposed similarity measures. We present the results alongside the median and maximum results of using bootstrap, to better understand if any of the similarity metrics achieves the best selection of dataset to bootstrap with. The results of these experiments are showcased in Table 6.13.

Discussion

There is a lot to unpack from the experiments in this section. We first started by comparing how each dataset behaves when bootstrapped with each other. From the analysis of Figs. 6.1 and 6.2, we conclude that there are variations with respect to the dataset that is used to perform the bootstrap with. We also noticed that some datasets seem to benefit more from bootstrap than others and, a particularly interesting insight, that the matrix is asymmetrical, meaning that what just because dataset *A* bootstraps well with dataset *B*, the inverse might not happen.

Table 6.11: Correlation between absolute differences of meta-features for all datasets and normalized gains. **Bolded** are the meta-features whose correlation values are the largest in absolute value.

Domain	Meta-Feature	Correlation to Normalized Gain
Landmark	best_node	-0.078
	elite_nn	-0.0056
	linear_discr	**-0.18**
	naive_bayes	0.043
	one_nn	-0.049
	random_node	-0.057
	worst_node	-0.054
Complexity	**C1**	**0.32**
	C2	**0.27**
	F3	**-0.38**
	L2	-0.038
	N1	0.004
	T2	-0.096
	T3	-0.054
	T4	0.0097

We presented a number of meta-features that were extracted from the datasets in question and calculated how they correlated with the normalized gains achieved by the bootstrapping experiments. Our analysis indicated that, while none of the meta-features exhibited correlations that we deem strong, we were able to select 4 meta-features that we considered to be worthy of further study. The selected meta-features encase two potential situations, one where minimizing their absolute difference should lead to good bootstrapping results ($linear_discr$ and $F3$), and another where maximizing their absolute difference should be considered ($C1$ and $C2$).

From the four meta-features chosen, we proposed four similarity metrics that were tested on the available experiments to decide which dataset to use for the bootstrapping process. The results showed that in more than half the cases, using one of the three similarity measures: *SM-c1*, *SM-c2*, and *SM-f3*, leads to better bootstrapping results than the median. Even if the similarity measures cannot produce the optimal choice for bootstrapping, considering the computational effort that is saved, we believe them to be a successful endeavor. We argue that we might be able to improve these results by combining two or more of the similarity metrics proposed for EDGE's bootstrapping mechanism. For example, a good candidate should be *SM-f3* together with *SM-c1* or *SM-c2*. It is also interesting to see the results for *SM-c1* and *SM-c2* when we acknowledge that they are effectively choosing the dataset to bootstrap with by maximizing the absolute differences in each of the meta-features' values.

We recognize that we might be biasing our choice of meta-features since we tested the similarity metrics on the same data that was used to calculate the correlation values. Nonetheless, we believe the approach taken has merit in setting the groundwork for further developments and tests to be conducted. The advantages of using the similarity metrics should also be more prominent

Table 6.12: Proposed similarity metrics. *Base* refers to the dataset that will be bootstrapped and *Boot* refers to the dataset whose learned models and graphs will be utilized to perform the bootstrapping. ε is a small constant, to avoid dividing by zero.

Meta-Feature	Proposed Similarity Formula	Proposed Nomenclature
linear_discr	$\lvert linear\_discr(Base) - linear\_discr(Boot)\rvert$	*SM-ld*
C1	$1/(\lvert C1(Base) - C1(Boot)\rvert + \varepsilon)$	*SM-c1*
C2	$1/(\lvert C2(Base) - C2(Boot)\rvert + \varepsilon)$	*SM-c2*
F3	$\lvert F3(Base) - F3(Boot)\rvert$	*SM-f3*

Table 6.13: Results for all Similarity Metrics proposed. **Bolded** values are the ones where using the metric produced better or equal values than the median of the results. Starred (*) results are then ones that achieved the maximum value. $\varepsilon = 0.01$.

Dataset	*SM-ld*	*SM-c1*	*SM-c2*	*SM-f3*	Median	Max.
ST-haberman	87.58	87.75	87.75	**88.40**	88.40	89.05
L9-haberman	88.56	88.40	88.40	88.56	88.73	89.54
ST-saheart	**87.55**	**88.20**	**88.20**	**87.99**	87.55	88.74
ST-led7digit	76.50	77.80	77.80	**78.30**	78.20	78.70
ST-balance	93.45	**94.09**	**94.09**	93.13	93.53	94.49
L9-pima	**90.62**	89.52	89.52	**90.62**	90.40	91.15
ST-mammographic	89.76	**89.82**	**89.82**	89.64	89.82	90.30
ST-vehicle	90.13	90.19	90.19	**90.66**	90.43	91.13
ST-german	89.40	**90.20**	**90.20**	**89.85**	89.85	90.35
ST-flare	**82.18**	**82.13**	**82.13**	81.89	81.92	82.27
H9-flare-F	**97.47**	**97.75***	**97.75***	**97.51**	97.42	97.75
ST-contraceptive	78.53	**79.17**	**79.17**	78.70	78.87	79.34
ST-yeast	82.75	**82.82**	**82.82**	82.31	82.77	83.25
L9-yeast1	**91.07***	89.39	89.39	90.01	90.06	91.07
ST-titanic	**78.13**	**78.13**	**78.13**	**78.13**	78.13	78.54
H9-abalone19	99.71	99.70	99.70	**99.72**	99.72	99.75
ST-yeast-1-4-5-8_vs_7	97.91	**98.05**	**98.05**	97.84	98.05	98.41
H9-winequality-red-4	98.45	**98.52**	**98.52**	**98.52**	98.46	98.60
H9-abalone-19_vs_10-11-12-13	**99.19**	**99.19**	**99.19**	**99.19**	99.19	99.26
ST-marketing	67.51	67.35	67.35	67.29	67.55	67.89
ST-cleveland	**84.73**	83.39	83.39	**84.73**	84.23	85.40

when the size of available previous runs increases in the database.

6.6 Final Remarks

After all the experiments, we are able to provide answers to the questions raised in Chapter 1 and detailed in Chapter 4. In Section 6.2 pitted our weight evolution approach against the initial implementation of EDGE, with ours performing better on 3 of the 4 datasets while lagging by a small margin in the fourth. In the same section, we took our approach and compared it with a baseline test suite consisting of the same models that composed EDGE's Reservoir. On most of the datasets our approach emerged victorious, such that we conclude that the weights of the graph ensembles can be evolved such that more powerful ensembles are produced.

Testing the bootstrapping approach, in Section 6.3, we came to the conclusion that using graphs and base models from previous runs of EDGE leads to improvement in the performance of the graph ensembles. Two degrees of bootstrapping were tested, using only the exploitation of the database of previous runs and balancing out the exploitation with the exploration of the Reservoir's models' configuration space. The results indicated that any degree of bootstrapping is almost always beneficial, improving on 19 out of the 21 datasets. For this reason, we answer positively on the question of whether EDGE can bootstrap itself of previous runs, on different datasets, successfully.

Inside Section 6.4, we took the liberty to explore how does EDGE's performance varies when we vary the degree of exploitation of the database of previous runs, as well as its population size. Overall, EDGE performs well with some degree of exploitation, and two trends emerge from the experiments. Except for the case of $PopSize = 20 \wedge Exploit = 50\%$, increasing either the population size or the exploitation value, an improvement in performance is observed. However, the improvement from increasing the exploitation value is slightly more significant. With respect to the datasets, EDGE seemed to perform well across the board, even when varying the number of samples, the number of features, or the number of target classes of a given dataset.

Finally, with regards to bootstrapping and the investigation of similarity metrics based on meta-feature extraction, we presented four different similarity metrics that would allow us to chose the dataset to bootstrap another with, in Section 6.5. While simulating their use on the available datasets yielded mixed results, with the best of them only improving on the median accuracy from bootstrapping in around 57% of the datasets, we are convinced that it is a viable path towards optimizing the dataset choice for bootstrapping. It is also curious to note that some datasets did not seem to benefit from using bootstrapping. As for the research question about choosing datasets to bootstrap with, without exhaustively searching all possible combinations, it can be done with reasonably satisfactory results. We encourage the use of the similarity metric $SM - f3$ based on the $F3$ measure, which is the Maximum Individual Feature Efficiency calculation.

In short, our novel improvements to EDGE take it several steps forward into a more powerful and capable tool.

Chapter 7

Conclusions

This thesis explores how the evolution of graph ensembles can be made to produce powerful ensembles. Our starting point is EDGE, a machine learning tool that evolves the topology of a population of graph ensembles. We argue that, with our proposals, we take it several leaps forward.

EDGE uses a Reservoir to fetch its base models that become nodes in the graph ensembles. In order to compare EDGE with a fair baseline, the same models and configurations used by the Reservoir were aggregated to form a baseline suite of models. Comparing the proposed weight evolution against the only results presented for the initial implementation of EDGE by its authors, our development improved on 3 of the 4 datasets by an average margin of 4.20 percentage points, while lagging behind in the other dataset by just 0.3 percentage points. Taking the comparison against our baseline suite of models, our approach managed to attain the best value for accuracy in 34 out of the 38 datasets, with gains as substantial as 30 percentage points. The bootstrap introduction was only tested on a smaller subset of the entire fleet of datasets, due to exhaustive testing of bootstrapping combinations. Nonetheless, the result was that out of 21 datasets tested, using the bootstrap technique yielded improvements on 19 of them, when compared to not using any kind of bootstrapping. The exhaustive search of bootstrap combinations allowed us to derive insights into what meta-features can be used as similarity metrics, in order to effectively choose datasets to bootstrap, instead of trying all possible combinations. Further study into the effects of EDGE's parameters in its overall performance lead to the conclusion that, while not always the case, in the general terms, it is better to use bigger populations sizes and some degree of bootstrapping. The population size, since it can more directly impact resource consumption, can be a good trade-off between resource consumption and predictive accuracy.

Taking notice of all the experiments made, we feel confident that all the research questions can be answered and somewhat close to our expectations. Introducing the evolution of weights in the graphs, alongside the topology updating, produces better ensembles than using just the topology. Our argument for this is that we can further specialize each node to the particular classes he is

better suited, since the trust each other node has on its predecessors, for each class, is evolved along the whole process. Even when the topology is changed, the nodes keep track of the trust its successors had in them, akin to a measure of merit. The bootstrapping mechanism allowed EDGE to exploit graphs and models that were learned from different data, but nonetheless, be useful in its evolution process. We consider this contribution to be especially relevant when we pair it to the similarity metric between datasets. We present the argument that this development allows one to never start a problem from scratch since we can always try to explore some similarities with other datasets to bootstrap our task. The fact that we can balance the exploitation of the bootstrap with the exploration of the configuration space for base models means that, while we exploit previously learned graphs, we do not let it limit our progress. It is also worth mentioning that the database of previous runs can be something that is shared among colleagues and continuously evolved to improve the results of the bootstrapping further. On the similarity metric setting, while we only made use of the proposed metric for EDGE's particular case, we hypothesize that it can be used in other settings with similarly good results.

Stemming from this thesis' work, one article was written and subsequently accepted in the 7th ICML Workshop on Automated Machine Learning (AutoML) [1], about the weight evolution to improve the classification performance of the evolution of graph ensembles [Fontes et al., 2020]. Another document is being written to a journal entitled Pattern Recognition Letters [2], in the context of pattern mining, specifically the experiments that were made in the context of bootstrap and meta-feature extraction.

Future Directions

While we consider EDGE to be fully developed at this point, there are always new ideas and new directions that can be leveraged. One of these is the support for regression tasks, in addition to the current classification capabilities. Supporting regression tasks should be a reasonable effort since the code base has already been developed with that future goal in mind. Nonetheless, metrics and prediction logic that are currently set for classification tasks need to be adapted to regression ones.

This work introduced a decision function to balance weight and topology changes. In the context of topology changes, we consider as a future direction the exploration of different operators on graphs, based on current literature of graph dynamics and evolution.

Our bootstrapping capabilities allowed EDGE to exploit previously learned graphs, but one more step that could be thought of is that of feedback on the part of EDGE based on how well the exploitation worked out in the end. This would entail heuristics to track the models and graphs that came from bootstrapping and understand how well they performed during the evolution process.

The extraction of meta-features from the dataset has been a rewarding task, but the thought remains that there might be a different approach to convert datasets to a common representation.

[1] https://sites.google.com/view/automl2020/home
[2] https://www.journals.elsevier.com/pattern-recognition-letters

The authors' suggestion is the experimentation with methods such as autoencoder networks and the usage of a latent space representation to construe the similarity metric.

Finally, since we are evolving a population of ensembles, it would be interesting to adapt EDGE for distributed computing platforms, leveraging the parallelization of certain aspects of EDGE, like the training of the weights of the ensemble and mutation of the graphs.

Bibliography

[Ai et al., 2019] Ai, S., Chakravorty, A., and Rong, C. (2019). Household power demand prediction using evolutionary ensemble neural network pool with multiple network structures. *Sensors*, 19(3):721. DOI:10.3390/s19030721.

[Al-Sahaf et al., 2019] Al-Sahaf, H., Bi, Y., Chen, Q., Lensen, A., Mei, Y., Sun, Y., Tran, B., Xue, B., and Zhang, M. (2019). A survey on evolutionary machine learning. *Journal of the Royal Society of New Zealand*, 49(2):205–228.

[Alamdari and Mehrabian, 2012] Alamdari, S. and Mehrabian, A. (2012). On a DAG partitioning problem. In *Proceedings of the 9th International Workshop in Algorithms and Models for the Web Graph (WAW'12), June 22-23 2012, Halifax, NS, Canada*, volume 7323 of *Lecture Notes in Computer Science*, pages 17–28. Springer. DOI:10.1007/978-3-642-30541-2_2.

[Alcalá-Fdez et al., 2011] Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., and Herrera, F. (2011). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17:255–287.

[Alfaro et al., 2018] Alfaro, E., Gámez, M., and García, N. (2018). *Ensemble Classifiers Methods*, chapter 3, pages 31–50. John Wiley & Sons, Ltd.

[Antonio and Coello, 2018] Antonio, L. M. and Coello, C. A. C. (2018). Coevolutionary multiobjective evolutionary algorithms: Survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 22(6):851–865. DOI:10.1109/TEVC.2017.2767023.

[Bäck, 1996] Bäck, T. (1996). *Evolutionary algorithms in theory and practice - evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press.

[Bäck et al., 2000] Bäck, T., Fogel, D., and Michalewicz, Z. (2000). Introduction to evolutionary algorithms. *Evolutionary computation*, 1:59–63.

[Bäck et al., 2018] Bäck, T., Fogel, D. B., and Michalewicz, Z. (2018). *Evolutionary computation 1: Basic algorithms and operators*. CRC press, 1 edition.

[Bäck et al., 1993] Bäck, T., Rudolph, G., and Schwefel, H.-P. (1993). Evolutionary programming and evolution strategies: Similarities and differences. In *Proceedings of the Second Annual Conference on Evolutionary Programming (EP'93)*. Citeseer.

[Breiman, 1997] Breiman, L. (1997). Arcing the edge. Technical report, Statistics Department, University of California at Berkeley, Berkeley, CA.

[Bui and Moon, 1996] Bui, T. N. and Moon, B. R. (1996). Genetic algorithm and graph partitioning. *IEEE Transactions on computers*, 45(7):841–855. DOI:10.1109/12.508322.

[Candanedo et al., 2017] Candanedo, L. M., Feldheim, V., and Deramaix, D. (2017). Data driven prediction models of energy use of appliances in a low-energy house. *Energy and Buildings*, 140:81–97.

[Cha, 2007] Cha, S.-H. (2007). Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1.

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In Krishnapuram, B., Shah, M., Smola, A. J., Aggarwal, C. C., Shen, D., and Rastogi, R., editors, *Proceedings of the 22nd ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD'16), August 13-17 2016, San Francisco, CA, USA*, pages 785–794. ACM. DOI:10.1145/2939672.2939785.

[Choi et al., 2010] Choi, S.-S., Cha, S.-H., and Tappert, C. C. (2010). A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, 8(1):43–48.

[Colonna et al., 2017] Colonna, G., Nakamura, E., Cristo, M., and Gordo, M. (2017). Anuran Calls (MFCCs) Data Set. https://archive.ics.uci.edu/ml (last seen on 2018/11/10).

[Conover, 1999] Conover, W. (1999). *Practical nonparametric statistics*. Wiley series in probability and mathematical statistics: Applied probability and statistics. Wiley.

[Darwin, 2009] Darwin, C. (2009). *The Origin of Species: By Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. Cambridge Library Collection - Darwin, Evolution and Genetics. Cambridge University Press, 6 edition.

[De la Peña Sarracén, 2017] De la Peña Sarracén, G. L. (2017). Ensembles of methods for tweet topic classification. In *Proceedings of the Second Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval'17) co-located with the 33th Conference of the Spanish Society for Natural Language Processing (SEPLN'17), September 19 2017, Murcia, Spain*, pages 15–19.

[Demšar, 2006] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30.

[Deng, 2012] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142.

[Dengsheng Zhang and Guojun Lu, 2003] Dengsheng Zhang and Guojun Lu (2003). Evaluation of similarity measurement for image retrieval. In *Proceedings of the International Conference on Neural Networks and Signal Processing, December 14-17 2003, Nanjing, China*, volume 2, pages 928–931 Vol.2. DOI:10.1109/ICNNSP.2003.1280752.

[Dietterich, 2000] Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Proceedings of the Multiple Classifier Systems, First International Workshop, (MCS'00), June 21-23 2000, Cagliari, Italy*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer.

[Elbeltagi et al., 2005] Elbeltagi, E., Hegazy, T., and Grierson, D. (2005). Comparison among five evolutionary-based optimization algorithms. *Advanced engineering informatics*, 19(1):43–53. DOI:10.1016/j.aei.2005.01.004.

[Finn et al., 2017] Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning (ICML'17), August 6-11 2017, Sydney, NSW, Australia*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

[Fleurent and Ferland, 1996] Fleurent, C. and Ferland, J. A. (1996). Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*, 63(3):437–461. DOI:10.1007/BF02125407.

[Fontes and Silva, 2019] Fontes, X. and Silva, D. C. (2019). EDGE: Evolutionary directed graph ensembles. *International Journal of Hybrid Intelligent Systems*, 15(4):243–256. DOI:10.3233/HIS-190273.

[Fontes et al., 2020] Fontes, X., Silva, D. C., and Abreu, P. H. (2020). W-EDGE: Weight updating in directed graph ensembles to improve classification. In *Proceedings of the 7th International Workshop on Automated Machine Learning (AutoML'20) collocated with the 37th International Conference on Machine Learning (ICML'20), July 18 2020*. Accepted.

[Galar et al., 2013] Galar, M., Fernández, A., Barrenechea, E., and Herrera, F. (2013). Eusboost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition*, 46(12):3460–3471. DOI:10.1016/j.patcog.2013.05.006.

[Garcia et al., 2013] Garcia, L. P. F., de Carvalho, A. C. P. L. F., and Lorena, A. C. (2013). Noisy data set identification. In *Proceedings of the 8th International Conference on Hybrid Artificial Intelligent Systems (HAIS'13), September 11-13 2013, Salamanca, Spain*, volume 8073 of *Lecture Notes in Computer Science*, pages 629–638. Springer.

[Géron, 2019] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2 edition.

[Goshtasby, 2012] Goshtasby, A. A. (2012). Similarity and dissimilarity measures. In *Image registration*, pages 7–66. Springer.

[Gu et al., 2020] Gu, K., Zhang, Y., and Qiao, J. (2020). Ensemble meta learning for few-shot soot density recognition. *IEEE Transactions on Industrial Informatics*.

[Hagberg et al., 2008] Hagberg, A., Swart, P., and S Chult, D. (2008). Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

[Hamaker and Boggess, 2004] Hamaker, J. S. and Boggess, L. (2004). Non-euclidean distance measures in airs, an artificial immune classification system. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'04), June 19-23 2004, Portland, OR, USA*, pages 1067–1073. IEEE.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer series in statistics. Springer, 2 edition.

[Hendrickson and Kolda, 2000] Hendrickson, B. and Kolda, T. G. (2000). Graph partitioning models for parallel computing. *Parallel computing*, 26(12):1519–1534. DOI:10.1016/S0167-8191(00)00048-X.

[Herrmann et al., 2017] Herrmann, J., Kho, J., Uçar, B., Kaya, K., and Çatalyürek, Ü. V. (2017). Acyclic partitioning of large directed acyclic graphs. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID'17), May 14-17 2017, Madrid, Spain*, pages 371–380. IEEE Computer Society / ACM. DOI:10.1109/CCGRID.2017.101.

[Kachitvichyanukul, 2012] Kachitvichyanukul, V. (2012). Comparison of three evolutionary algorithms: Ga, pso, and de. *Industrial Engineering and Management Systems*, 11(3):215–223. DOI:10.7232/iems.2012.11.3.215.

[Kagie et al., 2009] Kagie, M., van Wezel, M., and Groenen, P. J. (2009). An empirical comparison of dissimilarity measures for recommender systems. *Erasmus Research Institute of Management (ERIM)*.

[Kim et al., 2002] Kim, Y., Street, W. N., and Menczer, F. (2002). Meta-evolutionary ensembles. In *Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN'02), May 12-17 2002, Honolulu, HI, USA*, volume 3, pages 2791–2796. IEEE. DOI:10.1109/IJCNN.2002.1007590.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15), May 7-9 2015, San Diego, CA, USA*, page 13.

[Kolodner, 1993] Kolodner, J. L. (1993). *Case-Based Reasoning*. Morgan Kaufmann.

[Lertampaiporn et al., 2012] Lertampaiporn, S., Thammarongtham, C., Nukoolkit, C., Kaewkamnerdpong, B., and Ruengjitchatchawalya, M. (2012). Heterogeneous ensemble approach with discriminative features and modified-smotebagging for pre-mirna classification. *Nucleic acids research*, 41(1):e21–e21. DOI:10.1093/nar/gks878.

[Lieberman et al., 2005] Lieberman, E., Hauert, C., and Nowak, M. A. (2005). Evolutionary dynamics on graphs. *Nature*, 433(7023):312–316. DOI:10.1038/nature03204.

[Lorena et al., 2019] Lorena, A. C., Garcia, L. P. F., Lehmann, J., de Souto, M. C. P., and Ho, T. K. (2019). How complex is your classification problem?: A survey on measuring classification complexity. *ACM Comput. Surv.*, 52(5):107:1–107:34.

[McVitie and Wilson, 1971] McVitie, D. G. and Wilson, L. B. (1971). The stable marriage problem. *Communications of the ACM*, 14(7):486–490. DOI:10.1145/362619.362631.

[Meir and Rätsch, 2003] Meir, R. and Rätsch, G. (2003). *An Introduction to Boosting and Leveraging*, pages 118–183. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Mendes-Moreira et al., 2012] Mendes-Moreira, J., Soares, C., Jorge, A. M., and Sousa, J. F. D. (2012). Ensemble approaches for regression: A survey. *Acm computing surveys (csur)*, 45(1):10. DOI:10.1145/2379776.2379786.

[Moreira et al., 2018] Moreira, O., Popp, M., and Schulz, C. (2018). Evolutionary multi-level acyclic graph partitioning. In Aguirre, H. E. and Takadama, K., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'18), July 15-19 2018, Kyoto, Japan*, pages 332–339. ACM. DOI:10.1145/3205455.3205464.

[Moyano et al., 2019] Moyano, J. M., Gibaja, E. L., Cios, K. J., and Ventura, S. (2019). An evolutionary approach to build ensembles of multi-label classifiers. *Information Fusion*, 50:168–180. DOI:10.1016/j.inffus.2018.11.013.

[Müller et al., 2016] Müller, A. C., Guido, S., et al. (2016). *Introduction to machine learning with Python: a guide for data scientists*. O'Reilly Media.

[Nickel et al., 2016] Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E. (2016). A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33. DOI:10.1109/JPROC.2015.2483592.

[Oberbreckling and Rivas, 2019] Oberbreckling, R. J. and Rivas, L. E. (2019). Techniques for dataset similarity discovery. US Patent 10,445,062.

[Onan et al., 2017] Onan, A., Korukoğlu, S., and Bulut, H. (2017). A hybrid ensemble pruning approach based on consensus clustering and multi-objective evolutionary algorithm for sentiment classification. *Information Processing & Management*, 53(4):814–833. DOI:10.1016/j.ipm.2017.02.008.

[Opitz and Maclin, 1999] Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198. DOI:https://doi.org/10.1613/jair.614.

[Oza and Russell, 2001] Oza, N. C. and Russell, S. (2001). *Online ensemble learning*. PhD thesis, University of California, Berkeley.

[Parthasarathy and Ogihara, 2000] Parthasarathy, S. and Ogihara, M. (2000). Exploiting dataset similarity for distributed mining. In Rolim, J. D. P., editor, *Proceedings of the International Parallel and Distributed Processing Symposium Workshops (IPDPS'00), May 1-5 2000, Cancun, Mexico*, volume 1800 of *Lecture Notes in Computer Science*, pages 399–406. Springer.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

[Patwary et al., 2019] Patwary, M. A. K., Garg, S. K., and Kang, B. (2019). Window-based streaming graph partitioning algorithm. In *Proceedings of the Australasian Computer Science Week Multiconference (ACSW'19), January 29-31 2019, Sydney, NSW, Australia*, pages 51:1–51:10. ACM. DOI:10.1145/3290688.3290711.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Pfahringer et al., 2000] Pfahringer, B., Bensusan, H., and Giraud-Carrier, C. G. (2000). Meta-learning by landmarking various learning algorithms. In Langley, P., editor, *Proceedings of the 17th International Conference on Machine Learning (ICML'00), June 29 - July 2 2000, Stanford, CA, USA*, pages 743–750. Morgan Kaufmann.

[Polikar, 2012] Polikar, R. (2012). *Ensemble Learning*, pages 1–34. Springer US, Boston, MA.

[Rajaraman and Ullman, 2011] Rajaraman, A. and Ullman, J. D. (2011). *Mining of Massive Datasets*. Cambridge University Press.

[Reif et al., 2012] Reif, M., Shafait, F., and Dengel, A. (2012). Meta-learning for evolutionary parameter optimization of classifiers. *Mach. Learn.*, 87(3):357–380.

[Rivolli et al., 2018] Rivolli, A., Garcia, L. P. F., Soares, C., Vanschoren, J., and de Carvalho, A. C. P. L. F. (2018). Towards reproducible empirical research in meta-learning. *CoRR*, abs/1808.10406.

[Runkler, 2016] Runkler, T. A. (2016). *Data Analytics - Models and Algorithms for Intelligent Data Analysis, Second Edition*. Springer.

[Seni and Elder, 2010] Seni, G. and Elder, J. F. (2010). Ensemble methods in data mining: improving accuracy through combining predictions. *Synthesis lectures on data mining and knowledge discovery*, 2(1):1–126. DOI:10.2200/S00240ED1V01Y200912DMK002.

[Stanley and Miikkulainen, 2002] Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127. DOI:10.1162/106365602320169811.

[Tettamanzi, 1998] Tettamanzi, A. G. B. (1998). Drawing graphs with evolutionary algorithms. In Parmee, I. C., editor, *Adaptive Computing in Design and Manufacture*, pages 325–337. Springer London.

[Thulasiraman and Swamy, 2011] Thulasiraman, K. and Swamy, M. N. (2011). *Graphs: theory and algorithms*. John Wiley & Sons.

[Van Rossum and Drake, 2009] Van Rossum, G. and Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

[Vilalta and Drissi, 2002] Vilalta, R. and Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artif. Intell. Rev.*, 18(2):77–95.

[Wang and Wang, 2006] Wang, X. and Wang, H. (2006). Classification by evolutionary ensembles. *Pattern Recognition*, 39(4):595–607. DOI:10.1016/j.patcog.2005.09.016.

[Webb and Zheng, 2004] Webb, G. I. and Zheng, Z. (2004). Multistrategy ensemble learning: Reducing error by combining ensemble learning techniques. *IEEE Transactions on Knowledge and Data Engineering*, 16(8):980–991. DOI:10.1109/TKDE.2004.29.

[Wilson, 2010] Wilson, R. J. (2010). *Introduction to graph theory*. Pearson, 5 edition.

[Wolpert et al., 1997] Wolpert, D. H., Macready, W. G., et al. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82. DOI:10.1109/4235.585893.

[Yang and Jin, 2006] Yang, L. and Jin, R. (2006). Distance metric learning: A comprehensive survey. *Michigan State Universiy*, 2(2):4.

[Yu and Gen, 2010] Yu, X. and Gen, M. (2010). *Introduction to evolutionary algorithms*. Springer Science & Business Media, 1 edition.

[Zhang and Ma, 2012] Zhang, C. and Ma, Y. (2012). *Ensemble machine learning: methods and applications*. Springer.

[Zhou et al., 2018] Zhou, T., Wang, S., and Bilmes, J. A. (2018). Diverse ensemble evolution: Curriculum data-model marriage. In *Advances in Neural Information Processing Systems 31*, pages 5905–5916. Curran Associates, Inc.

[Zhou, 2015] Zhou, Z. (2015). Ensemble learning. In Li, S. Z. and Jain, A. K., editors, *Encyclopedia of Biometrics, Second Edition*, pages 411–416. Springer US.

[Zitzler et al., 2000] Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195. DOI:10.1162/106365600568202.