



UNIVERSIDADE FEDERAL DE SANTA CATARINA  
CENTRO TECNOLÓGICO  
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E  
SISTEMAS

Vitor Hugo Medeiros De Luca

**Analysis of Machine Learning Ensembles Based on Diversity Measures**

Florianópolis, Brazil  
2020

Vitor Hugo Medeiros De Luca

**Analysis of Machine Learning Ensembles Based on Diversity Measures**

Thesis submitted to the Post-Graduation Program in Automation and Systems Engineering to obtain the Master's degree in Automation and Systems Engineering.

Advisor: Prof. Jomi Fred Hübner, Dr.

Florianópolis, Brazil

2020

Ficha de identificação da obra elaborada pelo autor,  
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

De Luca, Vitor Hugo Medeiros  
Analysis of Machine Learning Ensembles Based on  
Diversity Measures / Vitor Hugo Medeiros De Luca ;  
orientador, Jomi Fred Hübner, 2020 .  
69 p.

Dissertação (mestrado) - Universidade Federal de Santa  
Catarina, Centro Tecnológico, Programa de Pós-Graduação em  
Engenharia de Automação e Sistemas, Florianópolis, 2020 .

Inclui referências.

1. Engenharia de Automação e Sistemas. 2. Aprendizado de  
Máquina. 3. Ensemble. 4. Stacking. 5. Diversidade de  
Modelos. I. Hübner, Jomi Fred. II. Universidade Federal de  
Santa Catarina. Programa de Pós-Graduação em Engenharia de  
Automação e Sistemas. III. Título.

Vitor Hugo Medeiros De Luca

**Analysis of Machine Learning Ensembles Based on Diversity Measures**

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Eric Aislan Antonelo, Ph.D.  
Universidade Federal de Santa Catarina

Prof. Mauricio Edgar Stivanello, Dr.  
Instituto Federal de Santa Catarina

Prof. Ricardo Azambuja Silveira, Dr.  
Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de mestre em Engenharia de Automação e Sistemas.

---

Prof. Werner Kraus Junior, Dr.  
Coordenador do Programa

---

Prof. Jomi Fred Hübner, Dr.  
Orientador

Florianópolis, 02 de Março de 2020.

This thesis is dedicated to my parents, my siblings and  
my grandmother, Avelina.

## **ACKNOWLEDGEMENTS**

I would like to thank my family: my mother Fátima, my father Antônio, my brother Gustavo and my sister Natália. People I admire a lot and who are always encouraging and supporting me. They are true examples for me.

I would like to thank professor Jomi for having accepted to guide me in this work, always being helpful and impeccable.

Finally, I would like to thank everyone who helped me, directly or indirectly, in this work. The professors of the Department of Automation and Systems and the colleagues of the course.

*"It is not the critic who counts; not the man who points out how the strong man stumbles, or where the doer of deeds could have done them better. The credit belongs to the man who is actually in the arena, whose face is marred by dust and sweat and blood; who strives valiantly; who errs, who comes short again and again, because there is no effort without error and shortcoming; but who does actually strive to do the deeds; who knows great enthusiasms, the great devotions; who spends himself in a worthy cause; who at the best knows in the end the triumph of high achievement, and who at the worst, if he fails, at least fails while daring greatly."  
(Theodore Roosevelt, 1910)*

## RESUMO

O aprendizado de máquina se popularizou muito nos últimos anos devido ao desenvolvimento de novos algoritmos e teorias, ao crescente acesso aos dados e à crescente capacidade computacional. Neste contexto, existe uma busca constante por maior exatidão das predições realizadas. Dentre as possibilidades existentes, o *ensemble* de vários modelos tem se destacado em competições de aprendizado de máquina e também em congressos da área por apresentar alta exatidão. No entanto, como consequência, alguns desafios surgem. A complexidade do modelo e o tempo de treinamento aumentam, enquanto a sua interpretabilidade diminui. Não é trivial determinar quais modelos devem ser utilizados e como eles devem ser organizados. Esta dissertação analisa o desempenho do *ensemble* e a sua relação com a diversidade dos modelos que o compõe com o objetivo de mostrar um conjunto de abordagens com boa probabilidade para se obter um bom resultado. Em outras palavras, fazer com que as predições do *ensemble* como um todo sejam melhores do que as predições individuais de qualquer um dos modelos que o compõe. Para que este objetivo fosse possível, foi construída uma arquitetura para realização dos experimentos, em que os algoritmos são executados de forma paralela. Os resultados mostram estratégias que funcionaram na prática baseados na literatura da área e no estudo da diversidade de modelos.

**Palavras-chave:** Aprendizado de Máquina, *Ensemble*, *Stacking*, Diversidade de Modelos, Computação Paralela



## RESUMO EXPANDIDO

### Introdução

O aprendizado de máquina se popularizou muito nos últimos anos devido ao desenvolvimento de novos algoritmos e teorias, ao crescente acesso aos dados e à crescente capacidade computacional. Neste contexto, existe uma busca constante por maior exatidão das predições realizadas. Dentre as possibilidades existentes, o *ensemble* de vários modelos tem se destacado em competições de aprendizado de máquina e também em congressos da área por apresentar alta exatidão. No entanto, como consequência, alguns desafios surgem:

- Não é trivial saber quais modelos melhoram o *ensemble* e como eles devem ser reunidos para obter melhores predições do que seus modelos individuais.
- Toda a complexidade do *ensemble* e a falta de interpretabilidade necessária para explicar o modelo aos reguladores e outras partes interessadas.
- O tempo para treinar e obter as respostas dos modelos aumenta consideravelmente, pois o *ensemble* exige uma grande quantidade de recursos computacionais.

Esta dissertação analisa o desempenho do *ensemble* e a sua relação com a diversidade dos modelos que o compõe com o objetivo de mostrar um conjunto de abordagens com boa probabilidade para se obter um bom resultado. Em outras palavras, fazer com que as predições do *ensemble* como um todo sejam melhores do que as predições individuais de qualquer um dos modelos que o compõe.

Para que este objetivo fosse possível, este trabalho propõe uma arquitetura de *ensemble* paralelo. Essa arquitetura visa facilitar a análise desses problemas, testando rapidamente abordagens diferentes, mostrando o impacto que cada modelo traz para o *ensemble* como um todo e, finalmente, como o *ensemble* pode ser aprimorado para trazer melhores predições.

### Objetivos

O principal objetivo desta dissertação é analisar o desempenho do *ensemble* e sua relação com a diversidade dos modelos que o compõem. Diferentes abordagens são analisadas, cada uma com diferentes tipos de diversidade, a fim de avaliar os ganhos de desempenho de cada uma.

Para alcançar o objetivo principal, o segundo objetivo é desenvolver uma arquitetura para executar o *ensemble* e suas tarefas subjacentes em paralelo. Essa arquitetura será usada como base de todas as análises aqui propostas, mostrando os impactos que cada modelo traz para todo o *ensemble* por meio de medidas de diversidade e desempenho.

### Metodologia

A metodologia utilizada é dividida inicialmente em duas partes principais. Primeiro, revisão da literatura sobre aprendizado de máquina, *ensemble*, *stacking* e diversidade de modelos. As principais publicações da área são revisadas de forma a compreender

melhor este campo, assim como o seu estado da arte. Segundo, revisão dos trabalhos realizados na área, suas tecnologias e arquiteturas, assim como as principais ferramentas disponíveis para que seja possível o desenvolvimento desta pesquisa.

Através destas duas partes iniciais, é feita a definição da arquitetura a ser utilizada na pesquisa, a seleção das abordagens de diversidade de modelos, assim como os dados a serem utilizados. Finalmente, os resultados obtidos são analisados de acordo com o embasamento teórico e apresentados, assim como as conclusões.

## **Resultados e Discussão**

Para chegar aos resultados, esta dissertação utiliza quatro abordagens principais de diversidade de *ensembles*. Cada uma delas é descrita a seguir.

Na primeira abordagem, dois modelos iguais com diferentes hiperparâmetros são utilizados. Conforme esperado, as métricas de diversidade foram ruins (baixa diversidade) e a melhoria trazida pelo *stacking* muito baixa, algo que já é esperado pela própria literatura.

Na segunda abordagem, vários modelos iguais com diferentes hiperparâmetros são testados. Estes hiperparâmetros são escolhidos procurando elevar a diversidade. Assim como ocorreu na primeira abordagem, as métricas de diversidade foram ruins e a melhoria trazida pelo *stacking* muito baixa. Isto levou a conclusão de que os hiperparâmetros por si só possuem pouca probabilidade de constituírem fonte de diversidade quando os modelos utilizados são iguais. Deve-se salientar que o modelo utilizado é o *gradient boosting*. Outros modelos poderiam ter resultados diferentes.

Na terceira abordagem, vários modelos diferentes entre si são utilizados. As métricas de diversidade mostraram um avanço significativo a partir deste ponto. Conforme esperado, os ganhos trazidos pelo *stacking* também são significativos, ou seja, o resultado final trazido pelo *stacking* é melhor do que os modelos que o compõem.

Na quarta abordagem, além de utilizar vários modelos diferentes, o dataset também é dividido aleatoriamente por colunas e atribuído a diferentes grupos de modelos para que esta divisão seja uma nova fonte de diversidade. O resultado é positivo e ainda melhor do que a terceira abordagem. As métricas de diversidade melhoraram ainda mais e, da mesma forma, o desempenho do *stacking*. Dentro desta abordagem, foi feita uma tentativa de melhorar os resultados usando alguns modelos da primeira camada do *stacking* na segunda camada. Esta tentativa não teve êxito (o *stacking* não trouxe melhores resultados), considerando as condições usadas na experiência realizada.

Por fim, ficou claro a relação entre diversidade de modelos e o desempenho do *stacking*. Existem ainda muitas outras formas de diversidade que são descritas como possibilidades nos trabalhos futuros mostrados no capítulo 6.

Um estudo do desempenho da arquitetura proposta também é realizado nesta etapa. Uma comparação é feita entre as abordagens paralela e sequencial. Ao final, fica comprovado que a abordagem paralela utilizada atinge tempos de treinamento melhores do que a abordagem sequencial, algo que é desejado ao implementar a abordagem paralela na arquitetura proposta.

## **Considerações Finais**

Este trabalho apresenta um objetivo bastante amplo e complexo: estudar o *ensemble* e a sua diversidade para mostrar um conjunto de abordagens que provavelmente alcançam um bom resultado. Em outras palavras, abordagens de *ensemble* que obtêm um resultado melhor do que qualquer um dos modelos individuais que o compõem.

Para atingir esse objetivo, é essencial o desenvolvimento de uma arquitetura que permita a elaboração rápida de experimentos com execução eficiente.

Em relação aos resultados obtidos, esta dissertação começa tentando comprovar os resultados já mencionados na literatura. Esta prova é obtida na prática, começando com o *stacking* de modelos semelhantes e com pouca diversidade. Os ganhos obtidos com o *stacking* são pequenos ou inexistentes. As experiências desenvolvidas depois disso evoluem no sentido de sempre aumentar a diversidade entre os modelos para buscar maiores ganhos com o *stacking*. No final, essa relação entre aumento da diversidade e melhoria trazida pelo *stacking* pôde ser observada. A primeira observação clara disso ocorre quando vários modelos diferentes são usados no *stacking*. As conclusões deste trabalho sobre desempenho do *stacking* são mostradas nos seguintes itens:

- Para obter melhorias de exatidão com o *stacking*, é muito importante usar modelos de diferentes naturezas, principalmente nas primeiras camadas.
- As medidas de diversidade são bons indicadores da capacidade que o *stacking* possui para melhorar o resultado final.
- Modelos com alta diversidade e baixa exatidão (modelos fracos) tornam mais fácil a obtenção de melhores resultados com o *stacking*.
- Os modelos de alta exatidão melhoram a exatidão final do *stacking*, mas ao mesmo tempo dificultam as melhorias trazidas pelo *stacking*. A introdução de modelos com alta exatidão no *stacking* deve sempre ser seguida pela introdução da diversidade, para que o *stacking* continue a trazer bons resultados.
- A maneira como o conjunto de dados é usado no *stacking* também pode trazer diversidade a ele. Estudar o conjunto de dados e as suas variáveis pode ser uma boa opção antes de iniciar a modelagem.
- Não existe uma receita pronta sobre como o *stacking* deve ser feito. Existem muitas variáveis em questão que podem alterar os resultados.

Com relação ao desempenho da arquitetura paralela proposta, os experimentos mostram um ganho quase cinco vezes em um processador de oito núcleos. Quanto maior o número de núcleos do processador, maior essa diferença se torna.

**Palavras-chave:** Aprendizado de Máquina. *Ensemble*. *Stacking*. Diversidade de Modelos. Computação Paralela.

## ABSTRACT

Machine learning has become very popular in recent years due to the development of new algorithms and theories, the increasing access to data and the increasing computational capacity. In this context, there is a constant search for greater accuracy in the predictions made. Among the existing possibilities, the ensemble of several models has stood out in machine learning competitions and also in congresses in the area for presenting high precision. However, as a consequence, some challenges arise. The complexity of the model and the training time increase, while its interpretability decreases. It is not trivial to determine which models should be used and how they should be organized. This thesis analyzes the performance of the ensemble and its relationship with the diversity of the models that compose it with the objective of showing a set of approaches with good probability to obtain a good result. In other words, making the predictions of the ensemble as a whole better than the individual predictions of any of the models that compose it. To make this objective possible, an architecture was built to carry out the experiments, where the models are executed in parallel. The results show strategies that have worked in practice based on the literature in the area and on the study of the diversity of models.

**Keywords:** Machine Learning, Ensemble, Stacking, Model Diversity, Parallel Computing

## LIST OF FIGURES

Figure 1 – Overfitting . . . . .	20
Figure 2 – Overfitting . . . . .	20
Figure 3 – Bootstrap aggregating steps. . . . .	26
Figure 4 – Stacking with 2 layers, using neural networks (NN), vector support machines (SVM) and linear regression (LR). . . . .	28
Figure 5 – Static part and its basic components. . . . .	33
Figure 6 – Stacking (left side) and its execution graph (right side) from a Node execution perspective. . . . .	34
Figure 7 – Intermediate “Dataset 3” receives data from both “(0, 0)” and “(0, 1)” Sublayers. . . . .	36
Figure 8 – “Dataset 3” and “Dataset 4” receiving data from different Sublayers. . . . .	37
Figure 9 – Intermediate “Dataset 3” training Dataset example. . . . .	37
Figure 10 – Example of not optimal solution brought by the the first solution. . . . .	38
Figure 11 – Example of training with synchronous and asynchronous parts. . . . .	39
Figure 12 – Example of training running all Nodes sequentially. The execution graph (right side) is seen under a Sublayer execution perspective. . . . .	40
Figure 13 – Example of prediction with synchronous and asynchronous parts. . . . .	42
Figure 14 – Stacking with two models in the Layer 0 and one model in the Layer 1. . . . .	48
Figure 15 – Stacking with 162 models in the Layer 0 and one model in the Layer 1 (not shown in this Figure). . . . .	50
Figure 16 – Stacking with 162 models in the Layer 0 and in the Layer 1. . . . .	52
Figure 17 – Eight different models in Layer 0 and in Layer 1. . . . .	53
Figure 18 – Three groups of five models in the Layer 0 plus five models composing the Layer 1. . . . .	57
Figure 19 – Eight different models in Layer 0 and in Layer 1. . . . .	59

## LIST OF TABLES

Table 2.1 – Confusion matrix for a binary classification problem. . . . .	20
Table 3.1 – Set of correct or wrong answers between two models. . . . .	29
Table 5.1 – Parameters used to generate the dataset. . . . .	47
Table 5.2 – Booster parameters used by the stacking models. . . . .	47
Table 5.3 – Metrics obtained for all models in the stacking. . . . .	48
Table 5.4 – Diversity measurements between the two models in the Layer 0. . . .	48
Table 5.5 – Range of hyperparams chosen to compose the Layer 0 of the stacking.	49
Table 5.6 – Parameters used to generate the dataset. . . . .	49
Table 5.7 – Metrics for all models in layer 0 and the model in the Layer 1 (1, 0, 0).	50
Table 5.8 – Diversity measurements for all models in the Layer 0. . . . .	51
Table 5.9 – Metrics for all models in the layer 1. . . . .	51
Table 5.10 – Diversity measurements between the models in the Layer 0. . . . .	54
Table 5.11 – Metrics for all models in the layer 0. . . . .	54
Table 5.12 – Metrics for all models in the layer 1. . . . .	54
Table 5.13 – Metrics for all models in the layer 1 without SVC in the Layer 0. . . .	55
Table 5.14 – Parameters used to generate the dataset. . . . .	56
Table 5.15 – Diversity measurements between all models in the Layer 0. . . . .	58
Table 5.16 – Metrics for each group in the Layer 0. . . . .	58
Table 5.17 – Metrics for all models in the layer 1. . . . .	59
Table 5.18 – Metrics for the group 4 models contained in the Layer 0. . . . .	60
Table 5.19 – Metrics for all models in the Layer 1. . . . .	60
Table 5.20 – Training user times running sequentially and in parallel. . . . .	61
Table 5.21 – Sequential and parallel times when running the algorithms with more than one thread. . . . .	61
Table A.1 – Parameters used to generate the dataset. . . . .	69
Table A.2 – Booster parameters used by the stacking models. . . . .	69
Table A.3 – Metrics for all models in layer 0 and the model in the Layer 1 (1, 0, 0).	69

## CONTENTS

<b>1</b>	<b>INTRODUCTION</b>	<b>16</b>
1.1	OBJECTIVES	17
1.2	CONSTRAINTS	17
1.3	OUTLINE	18
<b>2</b>	<b>SUPERVISED LEARNING</b>	<b>19</b>
2.1	OVERFITTING	19
2.2	CROSS-VALIDATION	19
2.3	PERFORMANCE MEASUREMENTS	20
<b>2.3.1</b>	<b>Accuracy</b>	<b>21</b>
<b>2.3.2</b>	<b>Precision</b>	<b>21</b>
<b>2.3.3</b>	<b>Recall</b>	<b>21</b>
<b>2.3.4</b>	<b>F1 score</b>	<b>22</b>
<b>3</b>	<b>ENSEMBLE LEARNING</b>	<b>23</b>
3.1	COMBINING OUTPUTS	23
<b>3.1.1</b>	<b>Majority vote</b>	<b>23</b>
<b>3.1.2</b>	<b>Averaging</b>	<b>24</b>
3.2	COMBINING BASE MODELS	25
<b>3.2.1</b>	<b>Bagging</b>	<b>25</b>
<b>3.2.2</b>	<b>Boosting</b>	<b>26</b>
<b>3.2.3</b>	<b>Stacking (Stacked Generalization)</b>	<b>27</b>
3.3	DIVERSITY BETWEEN MODELS	28
<b>3.3.1</b>	<b>Pairwise diversity measures</b>	<b>29</b>
3.3.1.1	Q statistic	29
3.3.1.2	Correlation coefficient $\rho$	30
3.3.1.3	Disagreement measure	30
3.3.1.4	Double-fault measure	30
<b>3.3.2</b>	<b>Non-pairwise diversity measures</b>	<b>30</b>
3.3.2.1	Entropy $E$	30
3.3.2.2	Kohavi-Wolpert variance	31
<b>4</b>	<b>PROPOSED ARCHITECTURE</b>	<b>32</b>
4.1	PARTS THAT MAKE UP THE ARCHITECTURE	32
<b>4.1.1</b>	<b>Static Part</b>	<b>32</b>
4.1.1.1	Layer	32
4.1.1.2	Sublayer	33
4.1.1.3	Node	33
<b>4.1.2</b>	<b>Dynamic Part</b>	<b>34</b>
4.1.2.1	Connections	34

4.1.2.2	Datasets . . . . .	35
4.1.2.2.1	<i>Input Datasets</i> . . . . .	35
4.1.2.2.2	<i>Intermediate Datasets</i> . . . . .	35
4.1.2.2.3	<i>Output Datasets</i> . . . . .	36
4.1.2.3	Cross-Validation . . . . .	36
4.2	TRAINING . . . . .	36
<b>4.2.1</b>	<b>First solution: run following the Layers execution order . . . . .</b>	<b>37</b>
<b>4.2.2</b>	<b>Second solution: run following the Datasets execution order . . . . .</b>	<b>38</b>
4.3	PREDICTION . . . . .	41
4.4	TOOLS USED IN THIS ARCHITECTURE . . . . .	41
4.5	PRACTICAL EXAMPLE . . . . .	43
<b>5</b>	<b>ENSEMBLE ARCHITECTURES . . . . .</b>	<b>45</b>
5.1	DATASET USED IN THE EXPERIMENTS . . . . .	45
5.2	FIRST APPROACH: TWO MODELS WITH DIFFERENT HYPERPARAMETERS . . . . .	47
5.3	SECOND APPROACH: MANY MODELS WITH DIFFERENT HYPERPARAMETERS . . . . .	49
5.4	THIRD APPROACH: DIFFERENT MODELS WITH LOW CORRELATION . . . . .	52
5.5	FOURTH APPROACH: HIGH DIVERSITY ENSEMBLES . . . . .	56
5.6	PARALLEL VS SEQUENTIAL COMPUTING PERFORMANCE DIFFERENCES . . . . .	60
<b>6</b>	<b>CONCLUSIONS . . . . .</b>	<b>63</b>
6.1	FUTURE WORK . . . . .	64
	<b>BIBLIOGRAPHY . . . . .</b>	<b>66</b>
	<b>APPENDIX A – ALTERNATIVE STACKING CONFIGURATION TO THE SECOND APPROACH . . . . .</b>	<b>69</b>



## 1 INTRODUCTION

Machine learning is an artificial intelligence field that studies the computational learning process. But what does “learning” mean in a computational context? One of the most remarkable definitions is given by Mitchell in (MITCHELL, 1997):

“A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ .”

*Tom M. Mitchell*

Russell and Norvig (RUSSELL; NORVIG, 2009) add that this experience is obtained through observations of the world. As an example, consider the task of classifying an email as spam or not. The task  $T$  is “classify emails”, the performance  $P$  could be the percentage of correctly classified emails, and the experience  $E$  could be a set of previously classified emails.

The way an algorithm gets experience from the world defines two major areas of machine learning. When the algorithm receives a set of observations (called “training set”) that already have the desired response (also called labels) for the task in question, the learning process is classified as supervised. When the algorithm needs to learn by itself in order to improve its performance, then the learning process is unsupervised.

This work is focused on supervised learning, where the algorithm must first learn from the received training set so that it can later predict observations that do not contain the expected response or output.

According to Jordan and Mitchell (JORDAN; MITCHELL, 2015), machine learning has progressed dramatically over the last two decades, becoming quite popular and gaining more importance. Among several factors that can be attributed to this growth, two stand out:

- Development of new machine learning algorithms and theories.
- Recent burst of online data and low-cost computing. In this context, we highlight the evolution of GPUs (Graphics Processing Unit) and cloud computing.

Machine learning competitions were also popularized. In 2009, the Netflix Prize<sup>1</sup> became very popular awarding US\$ 1M to the best solution. Hundreds of predictive models were put together to build the solution that reached the first place, however this solution became so complex that Netflix decided not to put it into production<sup>2</sup>.

One of the most well known competition platforms, Kaggle<sup>3</sup>, hosts hundreds of competitions. Observing the winning solutions of these competitions, it is possible to

<sup>1</sup> Competition official website: <https://www.netflixprize.com/>.

<sup>2</sup> <http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html>.

<sup>3</sup> Kaggle official website: <https://www.kaggle.com/>.

note that most of them are composed of not only a single learning algorithm, but an ensemble of them. The ensemble learning will be explained in more detail in chapter 3, but, in short, it consists of a set of predictive models organized so that they provide a single joint prediction.

As a consequence of the increased accuracy brought by the ensemble, some problems arise and are listed below.

- It is not trivial to know which algorithms fit the problem and how they should be assembled to get better predictions than its individual models.
- The whole ensemble complexity and the lack of interpretability necessary to explain the model to regulators and other stakeholders.
- The time to train and to get the responses from the models grows considerably as ensemble demands a large amount of computational resources.

The first item is directly linked to the objective of this work. This thesis analyzes the ensemble and its performance based on diversity measures that, according to the literature in the area, plays a fundamental role in the performance of the ensemble.

For this analysis to be possible, this work proposes a parallel ensemble architecture. This architecture aims to make it easier to analyze these issues by quickly assembling different approaches, showing the impact that each model brings to the complete ensemble and, finally, how the ensemble could be improved to bring better predictions. Using this architecture, this thesis aims to present a set of strategies to get good ensembles given some restrictions that will be considered throughout the work.

## 1.1 OBJECTIVES

The main objective of this thesis is to analyze the performance of the ensemble and its relationship with the diversity of the models that compose it. Different approaches are analyzed, each one with different types of diversity, in order to evaluate the performance gains of each one.

To achieve the main objective, the second objective is to develop an architecture to execute the ensemble and its underlying tasks in parallel. This architecture will be used as the basis of all analyses here proposed, showing the impacts that each model brings to the whole ensemble through measures of diversity and performance.

## 1.2 CONSTRAINTS

The objective presented in this chapter is quite broad and therefore it is necessary to establish some constraints in order to make clear the scope of this thesis. These constraints are listed below:

- All problems are binary classification.
- All datasets used have a target with balanced classes, that is, equally distributed.
- This thesis does not deal with big data issues and simulations can be reproduced in a personal computer. When this does not happen, the text will highlight this feature.
- The data used are structured and numerical. Data preprocessing steps are beyond the scope of this work.

### 1.3 OUTLINE

This thesis is structured according to the following chapters:

- Chapter 2 presents the basic concepts of supervised learning, introducing the nomenclature used throughout the document. Overfitting is presented as a brief introduction to the bias-variance tradeoff, as well as the cross-validation used to mitigate the problem.
- Chapter 3 introduces the concept of ensemble learning under two approaches. First, it considers only the outputs of the models. Then it considers the combination of weaker models to form a unique model.
- Chapter 4 presents the architecture here developed, all its definitions and the way the ensemble is built including training and predictions.
- Chapter 5 presents all the approaches studied for the ensemble and their corresponding results. It also presents a comparison between the parallel approach implemented versus a sequential approach.
- Chapter 6 presents the conclusions of this work. It also provides an overview of future work taking into account the gathered experience during the work.

## 2 SUPERVISED LEARNING

According to Russell and Norvig (RUSSELL; NORVIG, 2009), a supervised learning consists of the following problem:

Given a training set with  $N$  examples composed by input-output pairs

$$(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N) \quad (1)$$

where each  $y_j$  was generated by an unknown function  $y = f(x)$ , find the function  $h$  that approximates the true function  $f$ .

Russell and Norvig add that  $x$  and  $y$  can be any value, and even they do not have to be numbers. The function  $h$  is a hypothesis. Learning is a search through the whole space of possible hypotheses for one that will perform well, even considering new examples that were not in the set  $N$  but were in the  $f$  domain.

In practice,  $x_j$  is a vector composed of numbers (real and/or integers) and/or categorical variables (for example, red, yellow or green). It can also be a matrix in the case of images.

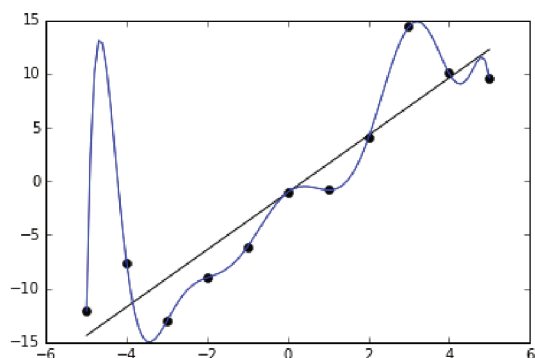
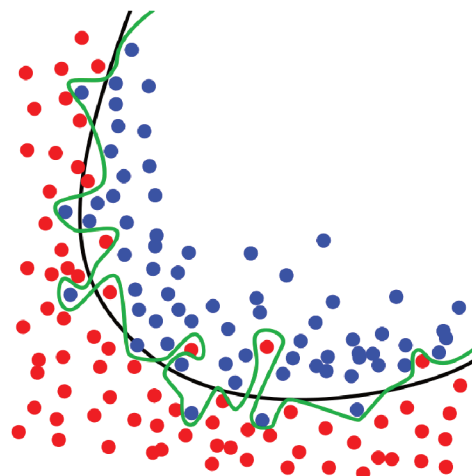
The output  $y_j$  can be a finite categorical variable. In this case, there is a classification problem. When there are only two categories, there is a binary classification problem. If the output  $y_j$  is a number, there is a regression problem.

### 2.1 OVERFITTING

To ensure that the hypothesis  $h$  (also called the model) has a good accuracy, it is evaluated on a test set with different examples from those contained in the training set. When a model has good accuracy over the training set, but poor accuracy over the test set, it means that it is not generalizing enough. In other words, there is an overfitting problem. The model is approximating too exactly the training set. Figures 1 and 2 show examples of this phenomenon. In Figure 1, the blue line is an example of overfitting, while the straight line has a greater generalization power. In Figure 2, the green line is an example of overfitting, while the black line has a greater generalization power.

### 2.2 CROSS-VALIDATION

There are several ways to reduce the likelihood of overfitting when training the model. One of the most used techniques is cross-validation. In this technique, the training set is divided into  $k$  parts, called  $k$ -folds. After this division, the model is trained on  $k-1$  parts and tested on the part that was left out of the training process. Then another part is chosen to stay out of the training process, performing the same process. After all parts have been used in this process in a circular way, the final model is the

Figure 1 – Overfitting<sup>1</sup>Figure 2 – Overfitting<sup>2</sup>

result of an operation on all the trained models. For example, an arithmetic mean of all of them.

### 2.3 PERFORMANCE MEASUREMENTS

The study of performance measurements is fundamental in machine learning. Note the definition of learning in a computational context presented in the introduction of this work and quoted by (MITCHELL, 1997) that brings the idea of “*performance measure P*”.

Each class of problems in machine learning has a set of metrics to be explored. As this thesis studies binary classification problems, only the metrics related to this class will be presented. The study presented by (FAWCETT, 2006) is a good reference in this area. Table 2.1 compares the predictions made by an algorithm against the actual known values in a supervised problem.

	Predicted class: true	Predicted class: false
True class: true	True Positive (TP)	False Negative (FN)
True class: false	False Positive (FP)	True Negative (TN)

Table 2.1 – Confusion matrix for a binary classification problem.

Each of these possibilities will be described below.

- True Positive (TP): it means that the prediction was correct. The correct value was “true” and the prediction obtained it.

<sup>1</sup> By Ghiles - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=47471056>.

<sup>2</sup> By Chabacano - Own work, GFDL, <https://commons.wikimedia.org/w/index.php?curid=3610704>.

- False Positive (FP): it means that the prediction was wrong. The correct value was “false” and the prediction obtained “true”.
- True Negative (TN): it means that the prediction was correct. The correct value was “false” and the prediction obtained this value.
- False Negative (FN): it means that the prediction was wrong. The correct value was “true” and the prediction obtained “false”.

TP and TN should be maximized because they are correct answers. However, depending on the type of problem, some errors (FP or FN) should be avoided more than others. Imagine diagnosing a disease. A FP error is better than a FN error. It is better to indicate that the patient is sick, even if he is not, and to oblige him to perform further exams than to indicate that the patient is not sick, if he is, avoiding his treatment.

Given these definitions, four types of performance measurements are presented below.

### 2.3.1 Accuracy

Accuracy is the sum of all correct predictions divided by the total of samples. As this thesis works only with balanced target problems, accuracy greater than 50% are better than random guessing. In problems with unbalanced target, high accuracy values must be compared with the proportion of the majority class. For example, 85% would not be a good result if the target's imbalance is 90% by 10%. Why? Because if all predictions were for the 90% class, then the accuracy would be 90% and this would not necessarily be a good result.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (2)$$

### 2.3.2 Precision

Precision is the ratio between the correct predictions for positive values (TP) and all predictions with positive values (TP + FP). In this way, precision can be seen as the correct prediction rate for predicted positive values. In the table 2.1, it is the rate present in the column "Predicted class: true".

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

### 2.3.3 Recall

Recall is the ratio between the correct predictions for positive values (TP) and all that should be in positive class (TP + FN). In this way, recall can be seen as the correct

prediction rate for all correct values. In the table 2.1, it is the rate present in the row "True class: true".

$$Recall = \frac{TP}{TP + FN} \quad (4)$$

#### 2.3.4 F1 score

F1 score is the harmonic mean between precision and recall.

$$F1\ score = \frac{2 * (Recall * Precision)}{(Recall + Precision)} \quad (5)$$

### 3 ENSEMBLE LEARNING

Ensemble learning will be presented in this chapter under two main approaches. First, only the outputs coming from different models will be considered, regardless of which model generated these outputs. Second, weaker base models are combined in order to generate a more accurate final model.

#### 3.1 COMBINING OUTPUTS

Among all the ways to combine the outputs from different models, two will be highlighted. The majority vote, used for classification problems, and averaging, used for regression problems.

##### 3.1.1 Majority vote

Suppose that there is a binary classification problem and that the unknown function  $y = f(x)$ , as proposed in (1), is always equal to 1. In other words, the correct answer is always 1. This binary classification problem is adapted from (VEEN; DAT; SEGNINI, 2015).

Now suppose that there are three models and that each of them has, individually, an accuracy of 0.75. That is, each model outputs the correct answer 75% of the time, missing the remaining 25%.

If a majority vote is used to combine these three models, then the following probabilities are expected:

1. All models are correct:

$$0.75 * 0.75 * 0.75 = 0.421875$$

2. Two models are correct, one is wrong:

$$\begin{aligned} &0.25 * 0.75 * 0.75 + \\ &0.75 * 0.25 * 0.75 + \\ &0.75 * 0.75 * 0.25 = 0.421875 \end{aligned}$$

3. Two models are wrong, one is correct:

$$\begin{aligned} &0.75 * 0.25 * 0.25 + \\ &0.25 * 0.75 * 0.25 + \\ &0.25 * 0.25 * 0.75 = 0.140625 \end{aligned}$$

4. All models are wrong:

$$0.25 * 0.25 * 0.25 = 0.015625$$



The sum of the first two scenarios (majority vote) leads to 0.84375 which is better than any of the individual models (0.75).

The formalization and generalization of this example is presented by (POLIKAR, 2006). Given  $T$  binary classification models whose outputs are statistically independent between each other, then the probability of a majority vote give the correct answer is given by:

$$P_{ens} = \sum_{k=(T/2)+1}^T \binom{T}{k} p^k (1-p)^{T-k} \quad (6)$$

$P_{ens}$  has a binomial distribution where  $k \geq (T/2)+1$  represents the majority vote. Applying 6 to the previous example would lead to the same 0.84375, using  $k = 2$ ,  $T = 3$  and  $p = 0.75$ .

The Condorcet Jury theorem (BOLAND, 1989) brings two important properties to 6:

$$\text{as } T \rightarrow \infty, \text{ then } P_{ens} \rightarrow 1, \text{ if } p > 0.5 \quad (7)$$

$$\text{as } T \rightarrow \infty, \text{ then } P_{ens} \rightarrow 0, \text{ if } p < 0.5 \quad (8)$$

It means that the more models are added, the better, if  $p > 0.5$ . In addition to these properties, it is important to notice that not all models should be necessarily chosen for voting. Major improvements can be achieved for majority voting when high diversity models are chosen. The meaning of “diversity” is quite broad and (BROWN; KUNCHEVA, 2010) brings a discussion on this topic. In practice, some correlation is calculated (e.g., Pearson correlation coefficient) in order to discover models with high correlation so that only one of them remains in the voting process.

The same problem can be tackled through weighted majority vote. In this case, a weight is given to each model according to its performance. As a consequence, the vote of a high performance model could only be changed by a group of models with low performance, indicating a possible mistake of this one.

### 3.1.2 Averaging

There are several ways to combine continuous outputs from various models (POLIKAR, 2006). The simplest of these is an arithmetic mean. When it is necessary to have an extra security for outliers values, an trimmed mean can be made, that is, maximum and minimum values are removed before performing the average. Another more advanced strategy is to perform a weighted average. When we know the weight of each model in the final response (there are several approaches to know this weight), then the weighted average is the best way to combine them. Finally, depending on the

models involved, it may be a good strategy to get the minimum, maximum, or even the median.

(POLIKAR, 2006) explains citing several references that it is not possible to know which of these approaches is the best because this choice depends on the structure of the data and prior knowledge about them.

## 3.2 COMBINING BASE MODELS

When it comes to combining models, there are three main techniques cited in the literature and widely used by the data science and machine learning community: bagging, boosting and stacking. They will be presented here.

### 3.2.1 Bagging

Bagging is the acronym for “Bootstrap aggregating” and was first introduced by Leo Breiman in 1996 (BREIMAN, 1996). Consider a learning set  $\mathcal{L}$  as proposed in (1) where  $y_n$  might be either a categorical or a numerical target variable. Consider also that there is a model that uses  $\mathcal{L}$  to create a predictor  $\varphi(x, \mathcal{L})$ . That is, a model that, given  $x$ , predicts  $y$  by  $\varphi(x, \mathcal{L})$ .

After presenting this scenario, Breiman proposes to generate  $k$  learning sets from  $\mathcal{L}$ , each one consisting of  $N$  bootstrap samples. This group of learning sets is called  $\{\mathcal{L}_k\}$ . Finally, the problem consists in using  $\{\mathcal{L}_k\}$  to create a better predictor than  $\varphi(x, \mathcal{L})$ , only using the group of models  $\varphi(x, \mathcal{L}_k)$ .

In order to solve this problem, the same article presents two options to aggregate the  $y$  outputs of all  $k$  models  $\varphi(x, \mathcal{L}_k)$ . First, when  $y$  is numerical, it is suggested to get the average of the outputs. Second, when  $y$  is categorical, it is suggested to perform a majority vote between them.

The name “Bootstrap aggregating” comes from these two mentioned procedures. First, “bootstrap” samples (random samples with replacement) to generate  $\{\mathcal{L}_k\}$  from  $\mathcal{L}$ . Second, “aggregating” the outputs from  $k$  models  $\varphi(x, \mathcal{L}_k)$ . The Figure 3 depicts these steps using decision trees as the base models.

Still, according to Breiman in (BREIMAN, 1996), bagging improves accuracy when the following models are used as base models:

- Neural nets;
- Classification and regression trees;
- Subset selection in linear regression.

An evolution of bagging called “Random forests” (RF) was presented by Breiman in 2001 (BREIMAN, 2001) using CART as its base models. It seeks to construct a highly

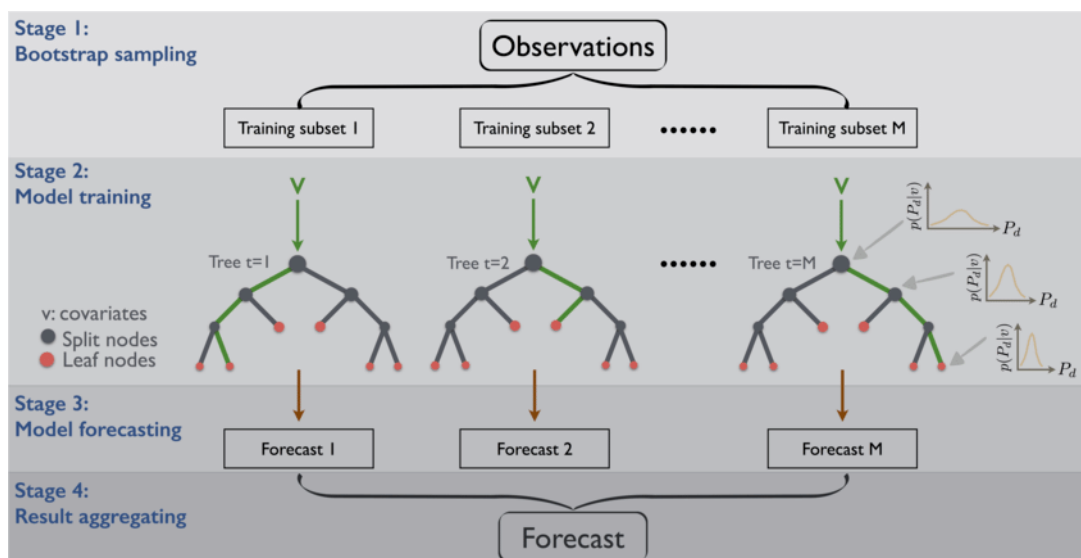


Figure 3 – Bootstrap aggregating steps (HE et al., 2016).

uncorrelated set of trees through the random selection of features to be used in the construction of each tree. This de-correlation between trees brings great improvements in accuracy.

An interesting property of bagging and RF is the ability to reduce variance. That is why they are used with decision trees. Because decision trees tend to have low bias and high variance. When used in the form of bagging or RF, they feature a powerful bias-variance tradeoff (HASTIE; TIBSHIRANI; FRIEDMAN, 2008).

### 3.2.2 Boosting

In an article called “The Strength of Weak Learnability” (1990) (SCHAPIRE, 1990), Robert E. Schapire answered a question raised by Kearns and Valiant (KEARNS, 1988) (KEARNS; VALIANT, 1989) if a set of weak learners, that is, a learner slightly better than a random guessing, could create a single strong learner, that is, a learner very close to the unknown function  $y = f(x)$ , as proposed in (1). This discussion ended up being the precursor of the boosting.

A few years later, in 1997, Schapire and Freund published the AdaBoost (Adaptive Boosting) algorithm (FREUND; SCHAPIRE, 1997) and won the 2003 Gödel Prize<sup>1</sup> for this. Friedman also brought a breakthrough for boosting with his “Gradient Boosting Machine” (FRIEDMAN, 2001) first published in 1999. Currently, the most powerful and known boosting algorithms are based on Friedman’s proposal. Among them, we highlight XGBoost (CHEN; GUESTRIN, 2016), LightGBM (KE et al., 2017), scikit-learn (PEDREGOSA et al., 2011), and gbm in R (RIDGEWAY, 2007).

Boosting is an algorithm that creates weaker models sequentially. There are

<sup>1</sup> <https://www.sigact.org/Prizes/Godel/>

many variations in the literature. The following algorithm provides an illustrative description of how it works (think of a regression problem):

1. Given the data set  $D = \{(x_i, y_i)\}$  with  $n$  examples,  $m$  features ( $x_i \in \mathbb{R}^m$ ) and  $y_i \in \mathbb{R}$ ;
2. The first model  $f_1(x)$  is trained on  $D$ ;
3. The predictions  $\hat{y}_i$  from  $f_1(x_i)$  are subtracted from the true values  $y_i$  generating the data set  $D_1 = \{(x_i, y_i - \hat{y}_i)\}$ . The features are the same, but the target variable is composed by the prediction errors of  $f_1(x)$ ;
4. The second model  $f_2(x)$  is trained on  $D_1$ ;
5. The same procedure performed in 3 applies here and this iteration remains until the error provided by the sum of the predictions of all the models  $f_k(x)$  is as low as desired.

Following these steps, the algorithm learns accurately the information contained in  $D$ , however it also has a great probability of overfitting. Boosting algorithms are known to be greedy. Therefore modern algorithms also implement a regularization in order to better generalize the model.

### 3.2.3 Stacking (Stacked Generalization)

The gains with ensemble begin to become really significant and close to the state of the art when techniques such as Stacking proposed by Wolpert (WOLPERT, 1992) are used. This proposal has an idea similar to neural networks, however, each node is represented by a machine learning algorithm (eg, logistic regression, gradient boosting, neural networks, among others). Like neural networks, learning occurs in several layers. To see how it works, Figure 4 illustrates a two-layer example. Consider that the original training set is divided to form two new sets: training and validation. The test set does not change. These are the three sets of the first layer.

When the learning begins, in the first layer, each algorithm is trained in the training set, making predictions on the validation set and on the test set. The prediction on the validation set generates a corresponding column for this algorithm in the training set of the second layer. Likewise, the prediction on the test set generates a corresponding column for this algorithm in the test set of the second layer. The union of these columns in the second layer to form the respective sets is the origin of the name "Stacking". Note that in the example there are three algorithms: neural networks (NN), support vector machines (SVM) and linear regression (LR). The Y response column of the validation set is replicated in the training set of the second layer. This is done because the training set of the second layer also needs to know the actual expected values.

		Train				Validation				Test			
		A	B	C	Y	A	B	C	Y	A	B	C	Y
1 <sup>st</sup> layer		0,56	0,59	1,24	1	2,51	0,32	5,10	0	1,51	0,54	3,15	?
		0,24	2,54	0,06	0	2,51	3,51	1,54	1	0,87	3,87	2,51	?
		5,12	2,21	0,84	0	4,12	2,10	2,41	0	2,21	1,31	1,01	?
		2,12	1,31	1,32	1	0,15	0,21	0,54	1	0,05	1,33	0,89	?
		0,11	2,31	0,54	1	2,74	0,51	4,65	1	1,91	0,55	1,71	?
		Train 2				Test 2							
		NN	SVM	LR	Y	NN	SVM	LR	Y				
2 <sup>nd</sup> layer		2,25	0,65	0,59	0	2,51	2,37	1,21	?				
		2,31	0,54	4,21	1	1,21	1,10	2,54	?				
		0,24	1,15	0,65	0	0,51	2,34	1,65	?				
		1,58	0,87	0,82	1	0,86	1,13	0,54	?				
		1,49	1,95	1,54	1	4,21	2,54	0,99	?				

Figure 4 – Stacking with 2 layers, using neural networks (NN), vector support machines (SVM) and linear regression (LR).

In the second layer, another machine learning algorithm will train on the training set of this layer to make predictions about the test set, generating the final prediction. This model in the second layer will probably have a higher accuracy when compared to other approaches like majority vote or arithmetic average between all the models outputs, among other simpler approaches.

An interesting property of Stacking is that it tends to improve when the correlation between the models that compose it is smaller. This is because each model has a different approach to the analyzed data. Stacking, when properly configured, is responsible for joining only the desirable characteristics of each model. In addition, the same model can be used multiple times with different data transformed into its inputs through feature engineering.

The downside of Stacking (and the ensemble as a whole) is that it consumes a lot of computational resources and also has a fairly high training time. Rarely this approach is used for commercial purposes, being restricted to the scientific community or machine learning competitions where the sole purpose is the final precision of the predictions.

### 3.3 DIVERSITY BETWEEN MODELS

When it comes to combining models, it does not matter if different models are being combined or if weaker base models are being combined to generate a single model, both situations require diversity between models. The greater the diversity between the models involved, the greater the gain generated by the ensemble of these models. The literature on this subject is quite extensive and unanimous. Some outstanding works in this area can be seen at (KUNCHEVA; WHITAKER, 2003), (TANG; SUGANTHAN;

YAO, 2006) and (BROWN; WYATT, et al., 2005).

In this thesis, it is necessary that diversity can be measured. The metrics used throughout this work will be those shown by (KUNCHEVA; WHITAKER, 2003) for being a well known reference in this area. All the diversity measures here presented were taken from this reference that can be consulted for more details.

Consider  $L$  to be the total number of classifiers. The output of each classifier  $D$  is represented by a  $N$ -dimensional binary vector. Also consider two classifiers,  $D_i$  and  $D_k$ . The possible responses between these two classifiers are shown in table 3.1.

	$D_k$ correct (1)	$D_k$ wrong (0)
$D_i$ correct (1)	$N^{11}$	$N^{10}$
$D_i$ wrong (0)	$N^{01}$	$N^{00}$

Table 3.1 – Set of correct or wrong answers between two models.

For example,  $N^{01}$  is the number of occurrences when  $D_i$  is wrong and  $D_k$  is correct. Consequently,  $N$  is equal to the number of true occurrences in  $N^{00} + N^{01} + N^{10} + N^{11}$ . Given these definitions, the diversity measures are divided into two major groups:

- Pairwise diversity measures: when metrics are calculated between pairs of models.
- Non-pairwise diversity measures: when metrics are calculated across all related models.

These groups are discussed in the following subsections.

### 3.3.1 Pairwise diversity measures

#### 3.3.1.1 Q statistic

The  $Q$  statistic between two classifiers  $D_i$  and  $D_j$  is shown in 9.

$$Q_{i,k} = \frac{N^{11} N^{00} - N^{01} N^{10}}{N^{11} N^{00} + N^{01} N^{10}} \quad (9)$$

Note that this value can vary between -1 (total disagreement) and 1 (total agreement). Classifiers that recognize the same samples as right or wrong tend to have positive values for  $Q$ . When there is a disagreement for the same samples, the value of  $Q$  tends to have negative values. This can be verified through the numerator of 9.

The average of all  $Q$  statistics is given by 10.

$$Q_{av} = \frac{2}{L(L-1)} \sum_{i=1}^{L-1} \sum_{k=i+1}^L Q_{i,k} \quad (10)$$

### 3.3.1.2 Correlation coefficient $\rho$

Correlation coefficient  $\rho$  is shown in 11. Both  $Q$  statistic and the correlation coefficient  $\rho$  always have the same sign because they have the same numerator. (KUNCHEVA; WHITAKER, 2003) mentions that it is possible to prove that  $|\rho| \leq |Q|$ .

$$\rho_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{\sqrt{(N^{11}N^{10})(N^{01}N^{00})(N^{11}N^{01})(N^{10}N^{00})}} \quad (11)$$

### 3.3.1.3 Disagreement measure

Disagreement measure is the sum of the disagreement ( $N^{01}$  and  $N^{10}$ ) divided by the total of samples.

$$Dis_{i,k} = \frac{N^{01} + N^{10}}{N^{11} + N^{10} + N^{01} + N^{00}} \quad (12)$$

### 3.3.1.4 Double-fault measure

Double-fault measure is the number of times the two models fail ( $N^{00}$ ) divided by the total of samples. In general, it is desirable that the models fail in different samples. This makes it easier for the ensemble to correct this error.

$$DF_{i,k} = \frac{N^{00}}{N^{11} + N^{10} + N^{01} + N^{00}} \quad (13)$$

## 3.3.2 Non-pairwise diversity measures

In order to understand non-pairwise diversity measures, a new definition needs to be made. Consider  $l(z_j)$  as the total number of classifiers among  $L$  that correctly classify a sample  $j$ .

### 3.3.2.1 Entropy $E$

For a given sample, maximum diversity occurs when approximately half of the classifiers ( $\lfloor L/2 \rfloor$ ) offer correct answers and the other half ( $L - \lfloor L/2 \rfloor$ ) offer wrong answers. Entropy  $E$  is shown in 14. It varies between 0 (minimum diversity) and 1 (maximum diversity).

$$E = \frac{1}{N} \sum_{j=1}^N \frac{1}{(L - \lfloor L/2 \rfloor)} \min\{l(z_j), L - l(z_j)\} \quad (14)$$

### 3.3.2.2 Kohavi-Wolpert variance

Kohavi-Wolpert variance is shown in 15. The higher this value, the greater the diversity.

$$KW = \frac{1}{NL^2} \sum_{j=1}^N l(z_j)(L-l(z_j)) \quad (15)$$

(KUNCHEVA; WHITAKER, 2003) also mentions that Kohavi-Wolpert variance is related to the averaged disagreement measure through this equation:

$$KW = \frac{L-1}{2L} Dis_{av} \quad (16)$$



## 4 PROPOSED ARCHITECTURE

This chapter describes the platform built for the development of this thesis. This platform sought to follow the original article proposed by Wolpert (WOLPERT, 1992), adding necessary elements to the most recent algorithms and also to the results to be presented in this work.

Although the platform is built base on stacking, it allows the study of various forms of ensemble besides stacking. That is why this thesis studies the ensemble as a whole and not just the stacking that could be considered a “subset” among all ensemble possibilities.

### 4.1 PARTS THAT MAKE UP THE ARCHITECTURE

In order to make clearer the components that make up the architecture, they will be divided into a static part that shows the main components of the ensemble and a dynamic part that shows the components related to the training and prediction execution.

#### 4.1.1 Static Part

The static part is thus defined because it is how the nodes (basic stacking units) are organized. This organization can be seen as a composition of these three basic units:

1. Layer
2. Sublayer
3. Node

Organized as follows:

- Layers are made up of sublayers.
- Sublayers are made up of Nodes.

These three basic units, as well as their relationships, are shown in Figure 5 and will be described below.

##### 4.1.1.1 Layer

Layer is the highest level stacking component. Each Layer consists of one or more Sublayers. Each Layer has a unique execution order and this order is always followed in a feedforward manner, i.e. each Layer only sends data to Layers with levels lower than its own. Similarly, no Layer sends data back to higher level Layers.

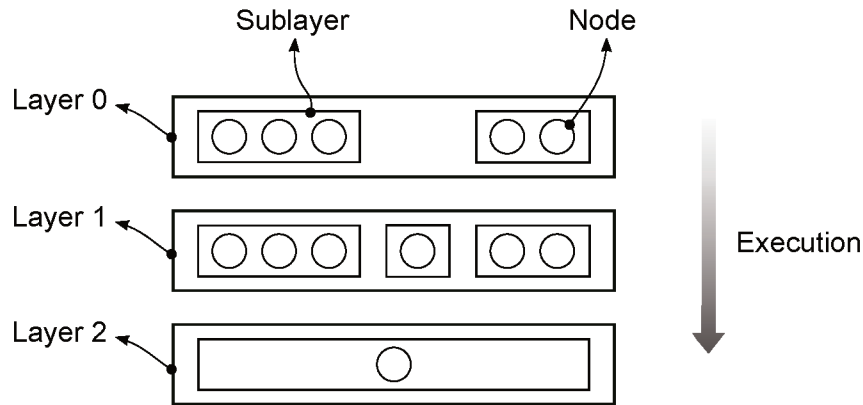


Figure 5 – Static part and its basic components.

#### 4.1.1.2 Sublayer

Sublayer is the second level stacking component and is made up of several nodes. Its nomenclature follows the pattern  $(x, y)$ , where “ $x$ ” is the Layer number where it is part of and “ $y$ ” is its identification number (a sequential number).

The Sublayers that make up a Layer run parallel to each other. This is the main, but not the only reason for Sublayer’s existence. Think about a scenario where Layers consist of Nodes directly.

In this context, it is important to highlight other features that reinforce the importance of a Sublayer:

- Each Sublayer has a single dataset common to all of its Nodes. This need will be further explained in the subsection 4.1.2.2.
- Sublayer is the basic unit for establishing the stacking execution flow. For example, Sublayer “(0, 1)” is linked to Sublayer “(1, 1)”. The meaning of this connection will be further explained in subsection 4.1.2.1. Similarly, it is not possible to establish connections between Layers or Nodes directly.

#### 4.1.1.3 Node

Node is the third and the most basic level in the proposed architecture. Its nomenclature follows the pattern  $(x, y, z)$ , where “ $x$ ” is the Layer where it is part of, “ $y$ ” is the Sublayer where it is part of and “ $z$ ” is its identification number which will also be its execution order. This is because, unlike Sublayers, the Nodes that make up a Sublayer run sequentially. This difference from Sublayers arises to be possible to configure synchronous and asynchronous parts when executing the training and prediction tasks. See the example in the Figure 6.

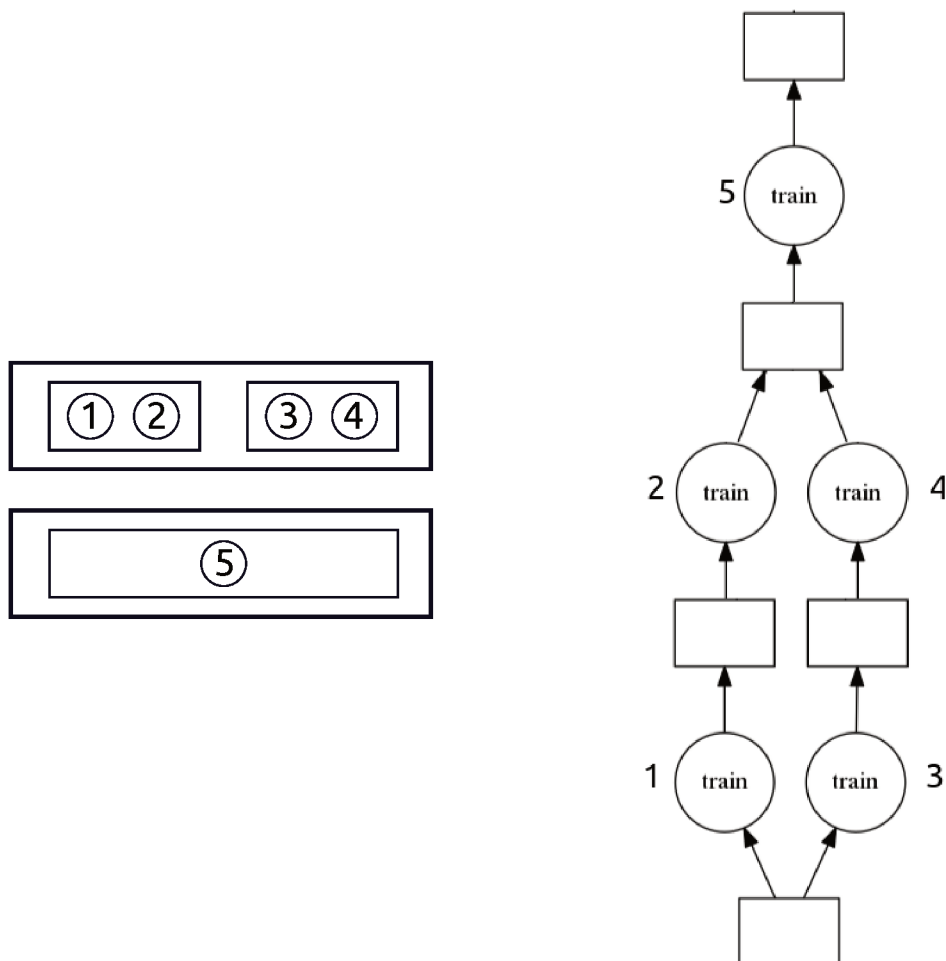


Figure 6 – Stacking (left side) and its execution graph (right side) from a Node execution perspective.

#### 4.1.2 Dynamic Part

The dynamic part is thus defined because it deals with how the ensemble will be executed. This includes the connection between Sublayers, datasets and cross validation. Each of these will be explained below.

##### 4.1.2.1 Connections

Connections determine ensemble data flow and always occur between Sublayers. This data flow includes both training data and test (prediction) data. They are determined in the form “((a, b), (c, d))”, where “(a, b)” refers to the source Sublayer and “(c, d)” refers to the destination Sublayer. Destination Sublayer is required to belong to lower level Layers and, if necessary, may skip some levels. For example, suppose three Layers 0, 1, and 2. The following conditions may occur:

- Any Sublayer belonging to Layer 0 can send data to Sublayers belonging to Layers 1 or 2.

- Sublayers belonging to Layer 1 can send data to Sublayers belonging to Layer 2, but not to Sublayers belonging to Layer 0.
- Sublayers belonging to Layer 2 can send data only to ensemble output.

#### 4.1.2.2 Datasets

Each Sublayer has an Dataset that is common to all Nodes contained in this Sublayer. More specifically, a Dataset consists of a pair containing a training dataset and a test dataset. Since these two datasets are always together, and for simplicity, only the word “Dataset” is mentioned. Datasets are divided into three groups: input, intermediate and output.

##### 4.1.2.2.1 Input Datasets

This is the ensemble input data and the required data for supervised learning. It must contain the training dataset with the desired target and the test (prediction) dataset without the target that have to be predicted. There may be more than one input Dataset in an ensemble. In the Figure 7, both “Dataset 1” and “Dataset 2” are input Datasets.

##### 4.1.2.2.2 Intermediate Datasets

Intermediate Datasets contains the data generated by each Sublayer’s Node. Depending on the configured connections, one intermediate Dataset may have data generated by more than one Sublayer. In the Figure 7, both “Dataset 3” and “Dataset 4” are intermediate Datasets. The “Dataset 3” is common to the two Sublayers of the Layer 1 because these two Sublayers receive the same data.

Figure 8 shows a different situation. There is a Dataset for each Sublayer of the Layer 1 (“Dataset 3” and “Dataset 4”) because they receive different data. The connections coming to them are now different.

Figure 9 shows the format of the training set that is part of the “Dataset 3” in the Figure 8. Note that the name of each column refers to each Layer 0 Node, as indicated by the connections. Note also the order of these columns. The Nodes “0\_0\_0” and “0\_0\_1” are synchronous because they are in the same Sublayer. The same happens with nodes “0\_1\_0” and “0\_1\_1”. However, since these two groups are in different Sublayers, they run in parallel and any order can occur between them. In practical terms, this parallel execution makes it necessary to control the reading and writing in the Datasets.

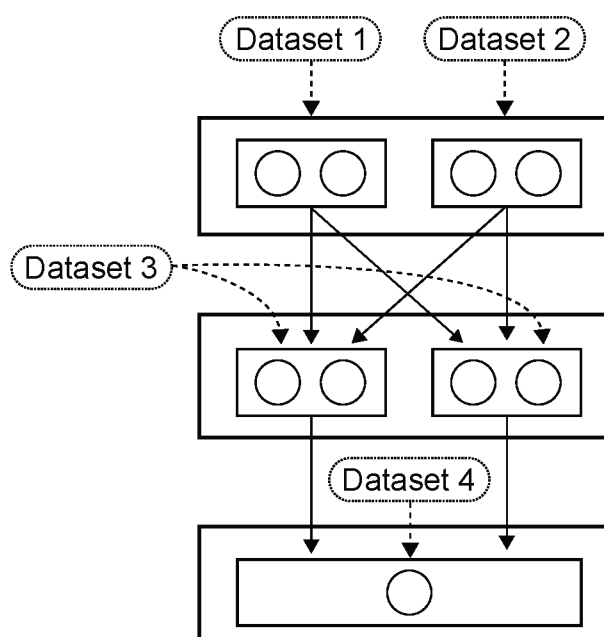


Figure 7 – Intermediate “Dataset 3” receives data from both “(0, 0)” and “(0, 1)” Sublayers.

#### 4.1.2.2.3 Output Datasets

There is an output Dataset that concentrates the ensemble’s final predictions as a whole. This Dataset does not concentrate training data as there is no need, only the final prediction values.

#### 4.1.2.3 Cross-Validation

Cross-validation is performed on a Dataset (input or intermediate Dataset) always prior to the training of any of the Nodes that perform training on this Dataset.

Cross-validation plays a key role in training, especially in heteroscedastic Datasets. The architecture presented here allows that each Node has a specific number of folds. However, the most common is that all Nodes that receive data from the same Dataset have the same number of folds so that a direct performance comparison can be made between them.

## 4.2 TRAINING

Performing training is one of the most complex steps in this architecture. It is not trivial to determine the correct execution order of the tasks considering all possible connections between Sublayers and that some parts are synchronous and others asynchronous.

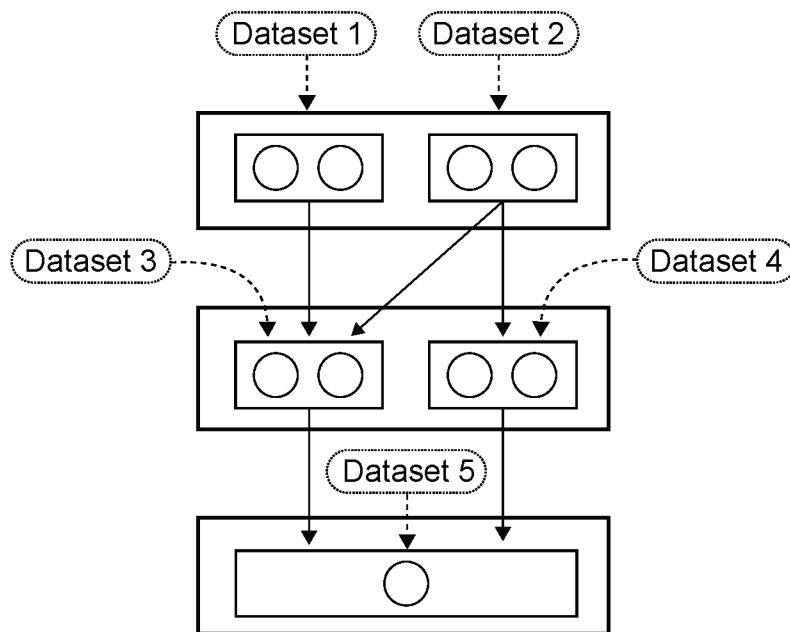


Figure 8 – “Dataset 3” and “Dataset 4” receiving data from different Sublayers.

	target	0_0_0	0_1_0	0_1_1	0_0_1
0	1	0.495750	0.789784	0.780785	0.807364
1	1	0.509465	0.772374	0.764384	0.786534
2	0	0.503128	0.892318	0.890990	0.979288
3	1	0.503128	0.793888	0.841302	0.939296
4	0	0.488752	0.181702	0.283118	0.103150

Figure 9 – Intermediate “Dataset 3” training Dataset example.

This thesis found two solutions that work considering the final result of the whole training. However, only the second solution is considered the best because it can parallelize all tasks following the best possible way.

#### 4.2.1 First solution: run following the Layers execution order

In this solution, the first Layer runs completely, followed by the second one, and so on. This solution is simple and works, but it is not optimal because during the execution of a Layer and its Sublayers, some Sublayers belonging to the next Layer may be ready to execute, but are not executed waiting for the current Layer to finish. The Figure 10 illustrates this possibility.

In the Figure 10, it would be possible to execute the Sublayer (1, 0) after the execution of the Sublayer (0, 0), but it is obliged to wait without need for the execution of the Sublayer (0, 1).

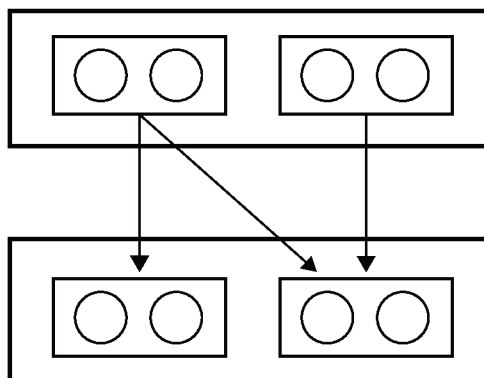


Figure 10 – Example of not optimal solution brought by the the first solution.

#### 4.2.2 Second solution: run following the Datasets execution order

In this solution, the execution sequence is determined by Datasets and not by Layers. If a Dataset is ready, then all Sublayers who train on this data are able to perform training. But what does a “ready Dataset” mean? It means that all Sublayers sending data to this Dataset have already finished execution. This section will describe each step of this solution in the order in which they are performed.

When the training is started, the first task is to create the intermediate Datasets avoiding redundancy between them. This task is described in detail in section 4.1.2.2.2.

Once the Datasets are configured, cross validation must be set. Each Dataset contains a set of Nodes by which it is the source of data. Since each Node has a number of folds specified for cross validation, each Dataset needs to prepare the required amount of folds for each Node. Nodes that have the same amount of folds will receive the same datasets (folds) so that there is a fair comparison between their results.

Following, the execution graph is dynamically defined and can have synchronous and asynchronous parts. This graph is defined based on the connections determined by the user for the ensemble. Figure 11 shows an example of four Nodes in four Sublayers in the first Layer followed by a Node in the second Layer. Figure 11 also shows the corresponding graph generated. Note the parallel execution of the four Nodes in the first Layer as they are in different Sublayers. The execution of the Node contained in the second Layer is performed after the execution of the four Nodes in the first Layer.

It is important to highlight in Figure 11 the presence of the tasks “pre\_train” and “pos\_train”. These tasks belong to each Dataset of the ensemble and are performed, as the name suggests, as a routine always before the training of the Sublayers that depend on this Dataset and afterwards as well.

Figure 12 shows an example of an ensemble similar to Figure 11, however, with four Nodes in a single Sublayer in the first Layer. Note that now only one training is performed in the first Layer precisely because there is only one Sublayer containing the four Nodes. These Nodes will perform training sequentially. The sequential execution of

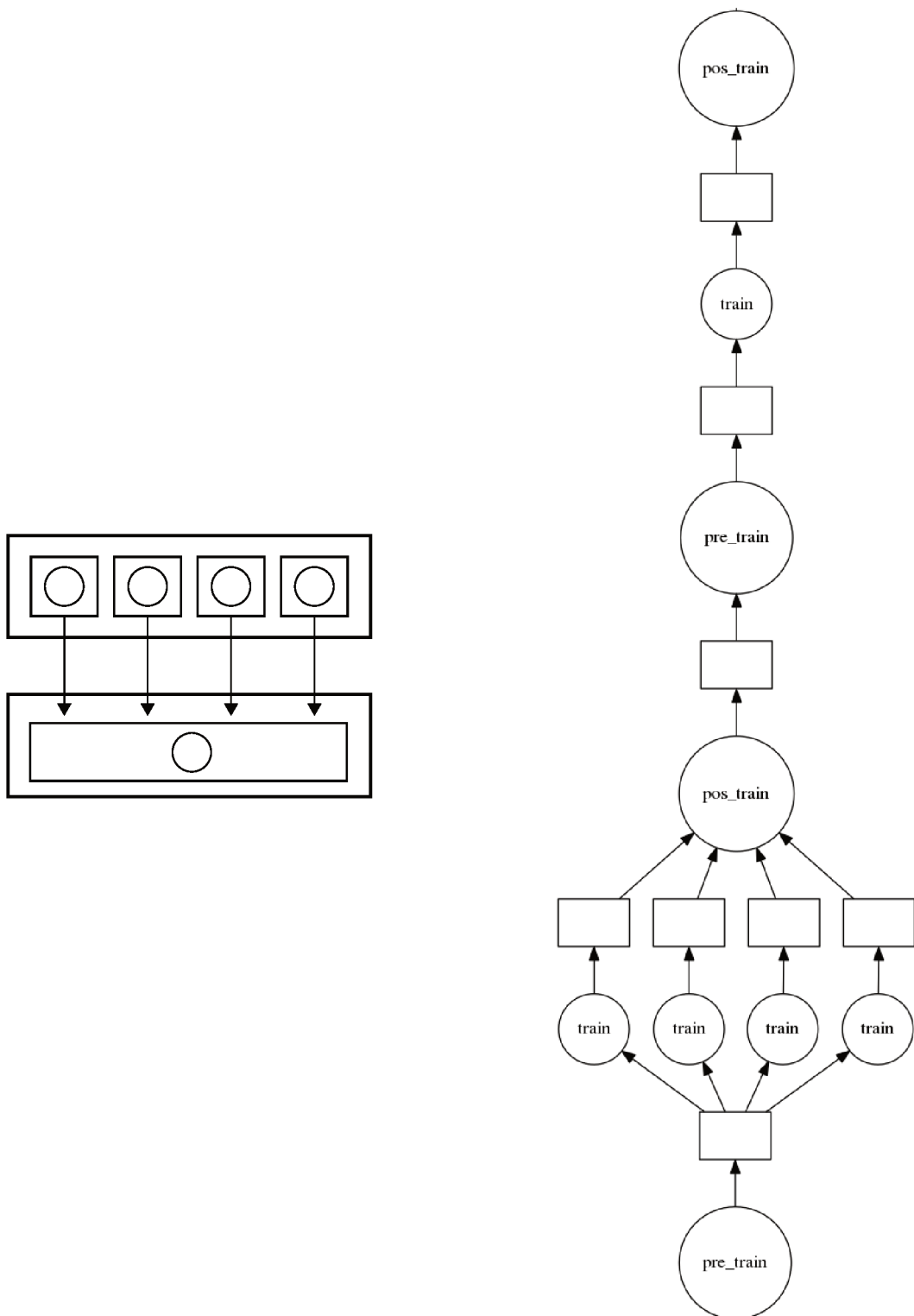


Figure 11 – Example of training with synchronous and asynchronous parts.



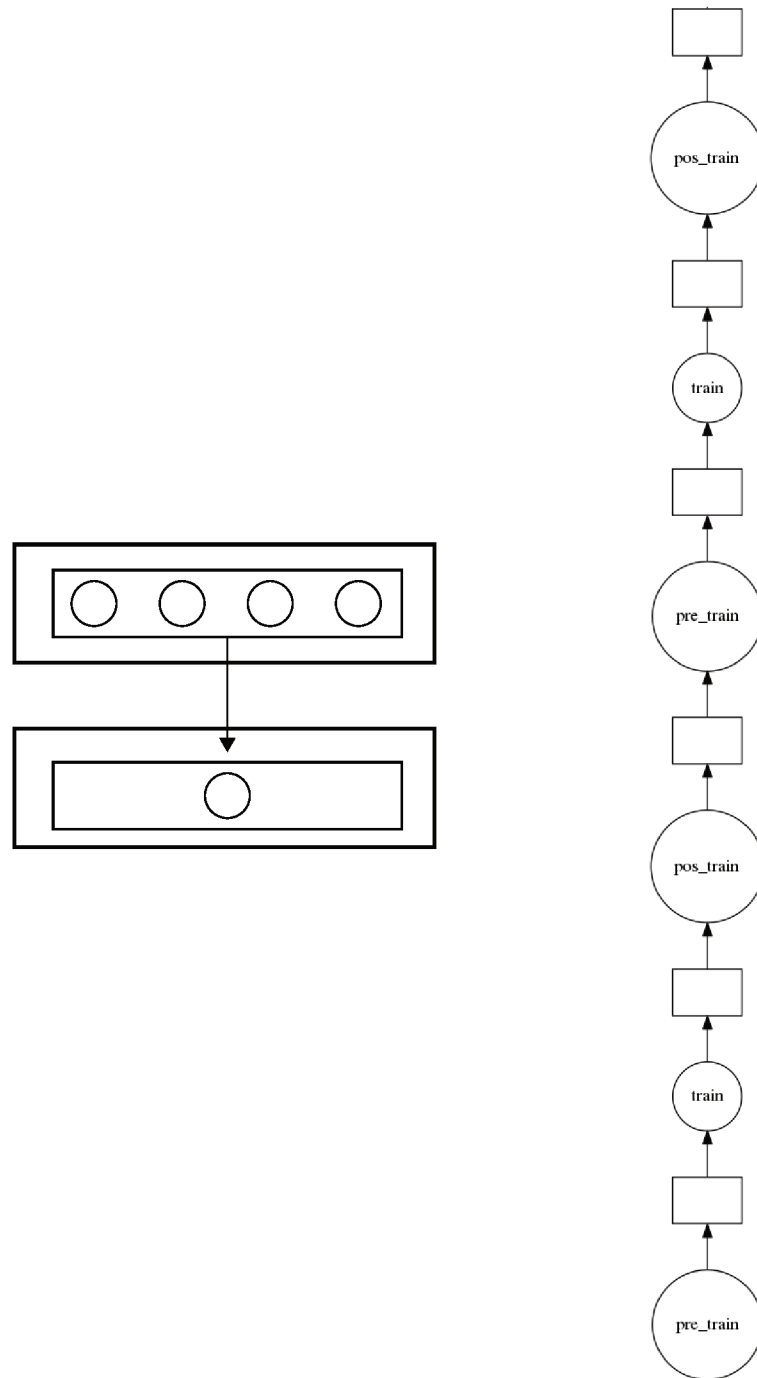


Figure 12 – Example of training running all Nodes sequentially. The execution graph (right side) is seen under a Sublayer execution perspective.

the four Nodes present in the first Layer is summarized in a single “train” task because Figure 12 shows the training execution graph under a Sublayer perspective (and so, it is not a Node perspective).

After defining the graph, the complete execution is triggered. Following the sequence provided by the graph, the following tasks happen for each Dataset:

- “pre\_train”: cross validation is prepared according to the desired number of folds.

- “train”: each Node that depends on this Dataset trains on n-1 folds and makes predictions on the holdout fold. These predictions are sent to the destination Dataset which is determined by the connections.
- “pos\_train”: destination Dataset is marked as “ready” and the Sublayers that depend on this Dataset are ready for execution.

After completing the whole execution, all Nodes are trained and the ensemble is ready to perform predictions.

### 4.3 PREDICTION

Predictions are made very similar to training with respect to the execution graph. Figure 13 illustrates a stacking example and its corresponding execution graph.

There is no need for cross validation for predictions. However, as cross validation was performed during training, so each fold generated a model. At the time of making predictions, each of these models will make predictions for the entire Dataset and a simple average between predictions is made to get the final result.

As with training, the final result is sent to the destination Dataset and will be used for next Layer predictions.

### 4.4 TOOLS USED IN THIS ARCHITECTURE

The entire architecture was developed using Python as it is one of the most used languages in data science and machine learning. In addition, some tools used by the architecture are described below:

- Dask<sup>1</sup> is open source and freely available software. It natively scales Python and provides parallelism for analytics. It is possible to scale up to clusters with its distributed scheduler and other support tools.
- Scikit-learn<sup>2</sup> is one of the most famous software libraries for machine learning in Python. It has several algorithms for classification, regression, clustering problems, among others.
- XGBoost<sup>3</sup> is a library for gradient boosting. It is widely used in machine learning competitions. (CHEN; GUESTRIN, 2016) is the article published by its authors.
- Catboost<sup>4</sup> is a library for gradient boosting with categorical features support.

<sup>1</sup> <https://dask.org/>

<sup>2</sup> <https://scikit-learn.org/stable/>

<sup>3</sup> <https://xgboost.readthedocs.io/en/latest/>

<sup>4</sup> <https://catboost.ai/>

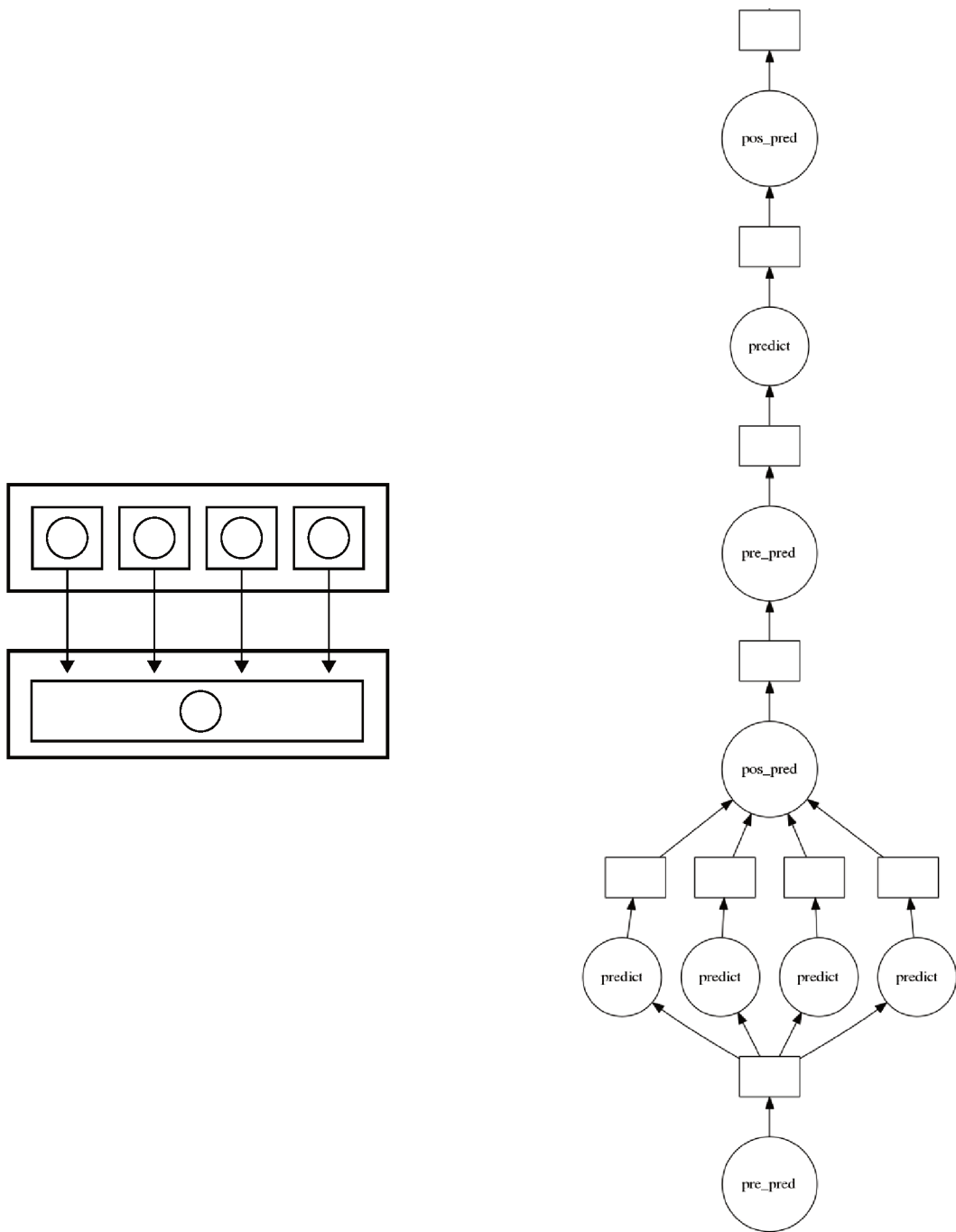


Figure 13 – Example of prediction with synchronous and asynchronous parts.

- Lightgbm<sup>5</sup> is a library for gradient boosting with fast training time among other important features.
- Graphviz<sup>6</sup> is a graph visualization software. It is open source.

#### 4.5 PRACTICAL EXAMPLE

In order to illustrate in a practical way how the architecture works, a pseudo code is shown below. In this code, three Nodes are inserted in Layer 0, in three different Sublayers to run in parallel. In addition, one Node is inserted in Layer 1. Each Node has the following algorithms:

- Node 0 (Layer 0, Sublayer 0): KNN (Scikit-learn).
- Node 1 (Layer 0, Sublayer 1): Gradient Boosting (LightGBM).
- Node 2 (Layer 0, Sublayer 2): SVC (Scikit-learn).
- Node 3 (Layer 1, Sublayer 0): Logistic Regression (Scikit-learn).

In order not to make the code too long, only LightGBM parameters are being shown.

---

```
# Start stacking
stacking = s.Stacking()

# Insert dataset as input to the Sublayers (0, 0), (0, 1) and (0, 2).
stacking.insert_dataset(train, "target", test, [(0, 0), (0, 1), (0, 2)])

# Insert KNN in Layer 0, Sublayer 0
node0 = stacking.insert_node('skl_knn', 0, 0)
# Insert lightGBM in Layer 0, Sublayer 1
node1 = stacking.insert_node('lightgbm', 0, 1)
# Insert SVC in Layer 1, Sublayer 0
node2 = stacking.insert_node('skl_svc', 0, 2)
# Insert Logistic Regression in Layer 1, Sublayer 1
node3 = stacking.insert_node('skl_logistic_regression', 1, 0)

# Configure model's params here as dictionaries.

# LightGBM params shown as example.
lgb_booster_params = {
    'learning_rate': 0.03,
    'max_depth': 5,
```

<sup>5</sup> <https://lightgbm.readthedocs.io/en/latest/>

<sup>6</sup> <https://www.graphviz.org/>

```
        'objective': 'binary',
        'metric': 'auc'
    }

lgb_params = {
    'params': lgb_booster_params,
    'early_stopping_rounds': 100,
}

# Config each Node with its params
node0.params = knn_params
node1.params = lgb_params
node2.params = svc_params
node3.params = lr_params

# Start training
stacking.train()

# Visualize training Stacking
stacking.train_exec_graph().visualize()

# Start predicting
predictions = stacking.predict()

# Visualize predicting Stacking
stacking.predict_exec_graph().visualize()

# See classification report for Node 0
node0_report = node0.classification_report(true_values)
print(node0_report)
```

---

## 5 ENSEMBLE ARCHITECTURES

Using the architecture described in the chapter 4, different approaches are analyzed in this chapter, each one with different types of diversity, in order to evaluate the performance gains of each one.

Diversity is already presented in the literature as one of the key properties for model stacking to produce better results than any of its individual models. In his own article on stacked generalization (WOLPERT, 1992), Wolpert already provides information in this regard.

“It is usually desirable that the level 0 generalizers are of all “types”, and not just simple variations of one another (e.g., we want surface-fitters, Turing-machine builders, statistical extrapolators, etc., etc.). In this way all possible ways of examining the learning set and trying to extrapolate from it are being exploited. This is part of what is meant by saying that the level 0 generalizers should “span the space”. Such spanning is important because stacked generalization isn’t just a way of determining which level 0 generalizer works best (as in cross-validation), nor even which linear combination of them works best (as in Gustafson et al.’s scheme); rather stacked generalization is a means of non-linearly combining generalizers to make a new generalizer, to try to optimally integrate what each of the original generalizers has to say about the learning set. The more each generalizer has to say (which isn’t duplicated in what the other generalizer’s have to say), the better the resultant stacked generalization.”

*David H. Wolpert*

Thus, diversity is the main reference for the results presented here. Initially, models with low diversity are intentionally tested so that later diversity is introduced and tested showing the improvements and the approaches that lead to these results.

### 5.1 DATASET USED IN THE EXPERIMENTS

The datasets used in this thesis are generated by scikit-learn<sup>1</sup>. It provides dataset generators for the most varied problems: classification, regression, clustering, multilabel classification, among many others. Below are the main reasons for using this type of dataset.

- Full control over all parameters of a dataset. These parameters are shown below in more detail.

<sup>1</sup> Scikit-learn website: <https://scikit-learn.org/stable/>

- The data provided is numeric. There is no need to the feature engineering step that is beyond the scope of this project.
- Flexibility and easy prototyping for necessary tests.
- Reproducibility guaranteed through the random state parameter needed to the random generation.

The way the software performs the creation of these datasets is described by its documentation<sup>2</sup>:

“This initially creates clusters of points normally distributed (std=1) about vertices of an  $n_{\text{informative}}$ -dimensional hypercube with sides of length  $2 * \text{class\_sep}$  and assigns an equal number of clusters to each class. It introduces interdependence between these features and adds various types of further noise to the data.”

Below, it is highlighted some of the most important parameters used by this thesis. They are described with more details in the make classification documentation<sup>2</sup>.

- Number of samples: number of rows to be generated.
- Number of features: total number of columns (features) to be generated.
- Number of informative features: how many features of the reported number of features are informative to the target.
- Number of redundant features: features constructed as linear combinations of informative features.
- Number of features repeated: these are randomly repeated features chosen from informative or redundant features.
- Number of classes: number of classes of the target. It is equal to two throughout the experiments.
- Number of clusters per class.
- Number of rows whose target is randomly assigned in order to introduce noise in the data. This makes the prediction task more difficult.
- Class separation: puts the clusters closer or further in order to make the separation between them more difficult.
- Random state: seed to be used by the random generator.

<sup>2</sup> Make classification website: [https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make\\_classification.html](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html)

## 5.2 FIRST APPROACH: TWO MODELS WITH DIFFERENT HYPERPARAMETERS

The first ensemble to be tested is simple in order to verify some properties already mentioned by the literature.

The model used is the gradient boosting implemented by the LightGBM framework (KE et al., 2017). This choice was made for two main reasons:

- The boosting algorithm has a large amount of hyperparameters to be modified and tested.
- LightGBM is an algorithm known for its good relationship between accuracy and speed of execution.

The characteristics of the dataset used are shown in the table.

Parameter	Value
Number of samples	50,000
Total number of features	20
Number of informative features	14
Number of redundant features	4
Number of repeated features	2
Fraction of target assigned randomly	0.15

Table 5.1 – Parameters used to generate the dataset.

Given this dataset created from the properties shown in the table 5.1, two models are inserted into the Layer 0 of the stacking ((0, 0, 0) and (0, 1, 0)) and another in the Layer 1 (1, 0, 0). Figure 14 illustrates this arrangement. Both models are gradient boosting with the booster params shown in table 5.2.

Booster parameter	Model (0, 0, 0)	Model (0, 1, 0)	Model (1, 0, 0)
Learning rate	0.1	0.03	0.03
Max tree depth	5	3	2
Metric	AUC	AUC	AUC
L1 regularization	0	0.1	0
L2 regularization	0	0.1	0
Minimum gain to split	0	0.1	0

Table 5.2 – Booster parameters used by the stacking models.

After executing this stacking using five folds across all datasets, the classification metrics obtained are shown in the table 5.3. The measures of diversity between the two Layer 0 models are shown in table 5.4.

The results are very consistent and are as expected. Note that the diversity measurements showed that the models have high similarity. Q statistics is very close to



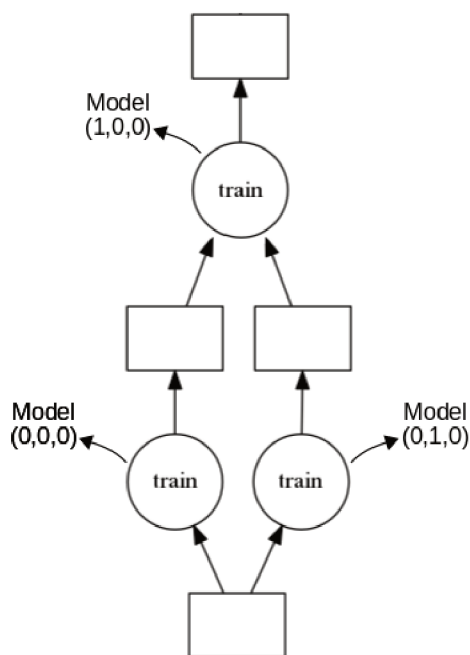


Figure 14 – Stacking with two models in the Layer 0 and one model in the Layer 1.

Metric	Model (0, 0, 0)	Model (0, 1, 0)	Model (1, 0, 0)
Precision - class 0	0.8717	0.8696	0.8742
Precision - class 1	0.8601	0.8582	0.8591
Recall - class 0	0.8613	0.8595	0.8598
Recall - class 1	0.8705	0.8683	0.8736
F1-score - class 0	0.8665	0.8645	0.8669
F1-score - class 1	0.8652	0.8632	0.8663
Accuracy	0.8659	0.8639	0.8666

Table 5.3 – Metrics obtained for all models in the stacking.

Diversity measure	Value
Q statistics	0.9945
Correlation coefficient $\rho$	0.8574
Disagreement measure	0.0333
Double-fault measure	0.1185
Entropy measure E	0.0333
Kohavi-Wolpert variance	0.0083

Table 5.4 – Diversity measurements between the two models in the Layer 0.

1, indicating that they recognize the same objects correctly (KUNCHEVA; WHITAKER, 2003). The disagreement measure is approximately 3.33%, a low value. Entropy also follows the same value.

Due to these diversity metrics shown, it is already expected that stacking does not bring a large accuracy gain compared to the individual models and this is what table

5.3 shows: model (1, 0, 0) registered a small accuracy gain over the models from Layer 0 ((0, 0, 0) and (0, 1, 0)). Many of the other metrics are also slightly better.

### 5.3 SECOND APPROACH: MANY MODELS WITH DIFFERENT HYPERPARAMETERS

Subsection 5.2 has been important to confirm some results mentioned in the literature. However, would it be possible to achieve high diversity using many different configurations (hyperparameters) over the same model? This subsection attempts to answer this question.

To perform this experiment, five gradient boosting hyperparameters were chosen to have their values changed. Each combination between these hyperparameters generates a model that is part of the Layer 0. These five hyperparameters were chosen seeking to cause the largest possible diversity between the models. Table 5.5 shows the range of values used for each hyperparameter.

Hyperparameter	Range of values
Learning rate	0.15, 0.25, 0.35
Max tree depth	4, 5, 6
L1 regularization	0, 0.1, 0.2
L2 regularization	0, 0.1, 0.2
Minimum gain to split	0, 0,1

Table 5.5 – Range of hyperparams chosen to compose the Layer 0 of the stacking.

The combination of these hyperparameters generates a total of 162 models. Each of them is placed in different Sublayers of the Layer 0 so that they run in parallel. Figure 15 illustrates this.

In addition to these 162 models inserted in Layer 0, another model (also gradient boosting) is inserted in Layer 1 for stacking and for getting the final result. This model in the Layer 1 is not shown in Figure 15. The parameters that generated the dataset for this experiment are shown in table 5.6.

Parameter	Value
Number of samples	50,000
Total number of features	20
Number of informative features	14
Number of redundant features	4
Number of repeated features	2
Fraction of target assigned randomly	0.15

Table 5.6 – Parameters used to generate the dataset.

The results of this experiment are similar to those found in the first approach shown in subsection 5.2. Even varying the hyperparameters, the main diversity mea-

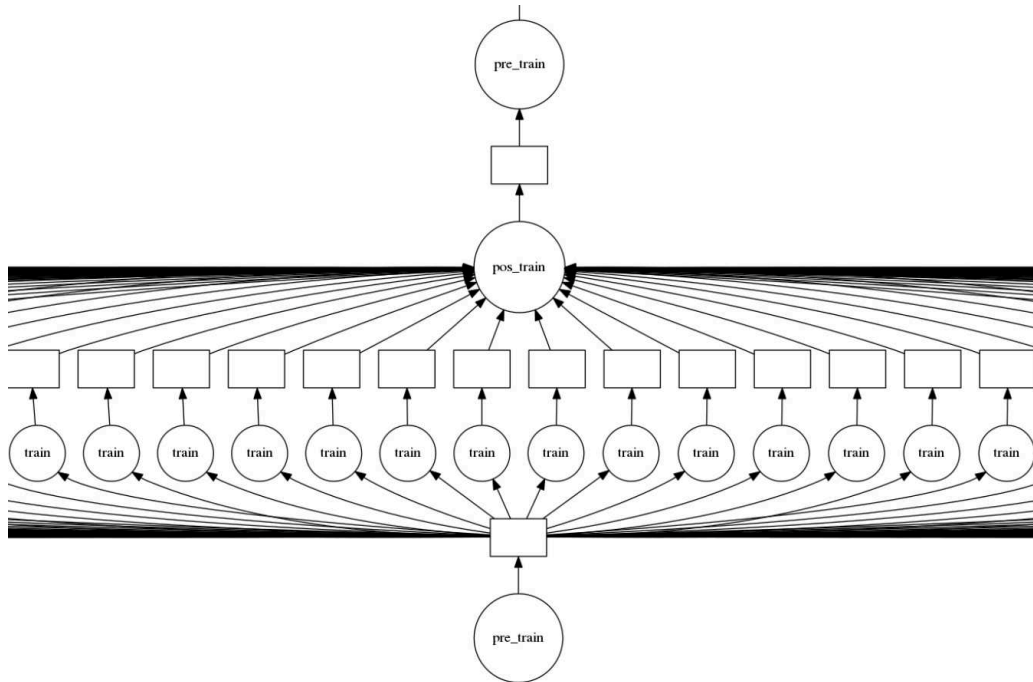


Figure 15 – Stacking with 162 models in the Layer 0 and one model in the Layer 1 (not shown in this Figure).

asures show little diversity between the models. Note that the disagreement measure, for example, has a maximum value of 0.02706 and an average value of 0.01358. Q statistics is very close to one. The best Layer 0 model achieved an accuracy of 0.9043 while the entire stacking output brought by the model (1, 0, 0) achieved an accuracy of 0.8982. In other words, the final accuracy is a little worse than the best Layer 0 model.

Metric	Layer 0 - Min	Layer 0 - Max	Layer 0 - Mean	Model (1, 0, 0)
Precision - class 0	0.892772	0.911284	0.904861	0.9006
Precision - class 1	0.883988	0.897886	0.892718	0.8958
Recall - class 0	0.885488	0.898681	0.893660	0.8976
Recall - class 1	0.891240	0.910782	0.904003	0.8988
F1-score - class 0	0.889124	0.904708	0.899225	0.8991
F1-score - class 1	0.887666	0.903956	0.898324	0.8973
Accuracy	0.888400	0.904333	0.898777	0.8982

Table 5.7 – Metrics for all models in layer 0 and the model in the Layer 1 (1, 0, 0).

This result could be considered worse than the experiment done in subsection 5.2 because the accuracy here got worse in the final stacking, however it can not be considered a setback. This result is unstable in the sense that small changes in both model hyperparameters and in dataset parameters can lead to a final stacking accuracy surrounding the best Layer 0 model. The important thing here is that the lack of diversity prevents a considerable advance in final stacking. Appendix A provides a configuration

of this experiment where the final stacking accuracy is slightly better than the best Layer 0 model (just to show that this is also possible).

Diversity measure	Min	Max	Average
Q statistics	0.99656	1.0	0.99899
Correlation coefficient $\rho$	0.85766	1.0	0.92594
Disagreement measure	0.0	0.02706	0.01358
Double-fault measure	0.08973	0.11033	0.09443
	Value		
Entropy measure E	0.01898		
Kohavi-Wolpert variance	0.00675		

Table 5.8 – Diversity measurements for all models in the Layer 0.

Continuing the experiment, there is still a possibility of questioning this result. As the dataset generated by the Layer 0 has 162 columns (one for each model), it is possible to question if the model present in Layer 1 is performing well. In other words, if the hyperparameters of this model are correctly adjusted. Otherwise, this misconfiguration could compromise all conclusions presented here as the final accuracy could be better.

Thinking about this possibility, a second experiment was performed. Just as multiple hyperparameters are tested on Layer 0, multiple hyperparameters are also tested on Layer 1. Figure 16 shows this stacking. Table 5.9 shows the results.

The hyperparameters are identical to those shown in table 5.5, except for the “Max tree depth” parameter which now has the range 9, 10 and 11 due to the greater number of features (163) in the dataset generated by the Layer 0 compared to the input dataset that has 20 features.

Metric	Layer 1 - Min	Layer 1 - Max	Layer 1 - Mean
Precision - class 0	0.900000	0.904794	0.902085
Precision - class 1	0.895194	0.900405	0.898359
Recall - class 0	0.896438	0.902639	0.900303
Recall - class 1	0.897709	0.903639	0.900169
F1-score - class 0	0.899584	0.902420	0.901192
F1-score - class 1	0.897596	0.900766	0.899262
Accuracy	0.898600	0.901600	0.900237

Table 5.9 – Metrics for all models in the layer 1.

The best Layer 1 model achieved an accuracy of 0.9016 which is better than the accuracy of the previously used model (0.8982), but still worse than the best Layer 0 model (0.9043). For diversity measures, the numbers showed even less diversity than Layer 0. For example, entropy reached 0.00736 and the double fault averaged 0.09648. Therefore, the conclusions presented here remain valid.

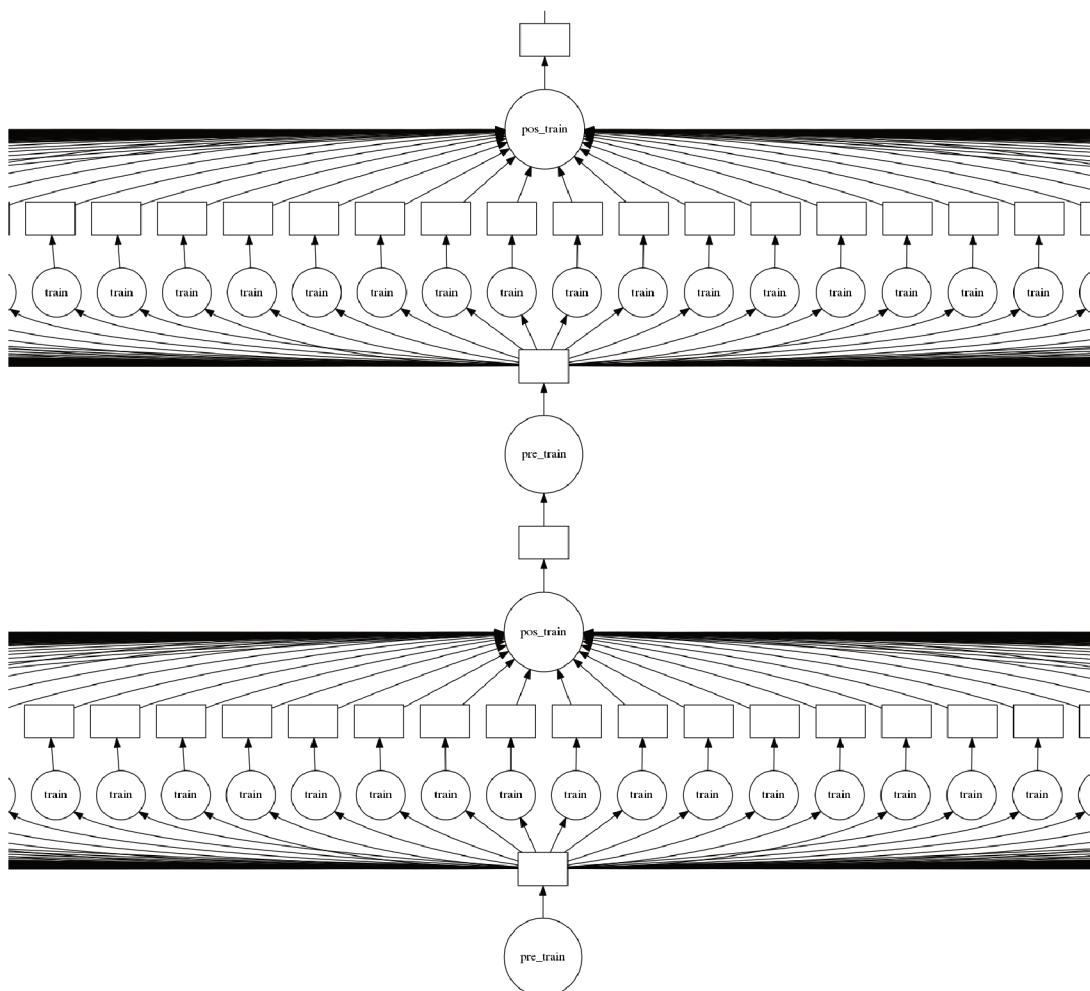


Figure 16 – Stacking with 162 models in the Layer 0 and in the Layer 1.

#### 5.4 THIRD APPROACH: DIFFERENT MODELS WITH LOW CORRELATION

This approach seeks to introduce diversity to the stacking through the use of different models. Due to the nature with which each model conducts the training, diversity is introduced and the results are presented here.

For this experiment, eight different models are chosen for Layer 0. For Layer 1, a single model could be used, but this model could compromise the results in the same way as occurred in section 5.3. Thus, eight different models are also inserted in Layer 1. The complete view of this stacking is shown in Figure 17.

The dataset used in this approach has two modifications compared to the previous experiment in the subsection 5.3 in order to make the prediction task more difficult. These modifications are as follows:

- Number of cluster per class: equal to 4 per class instead of 2.
- Class separation: equal to 0.8 instead of 1. This puts the clusters closer and the separation between them gets more difficult.

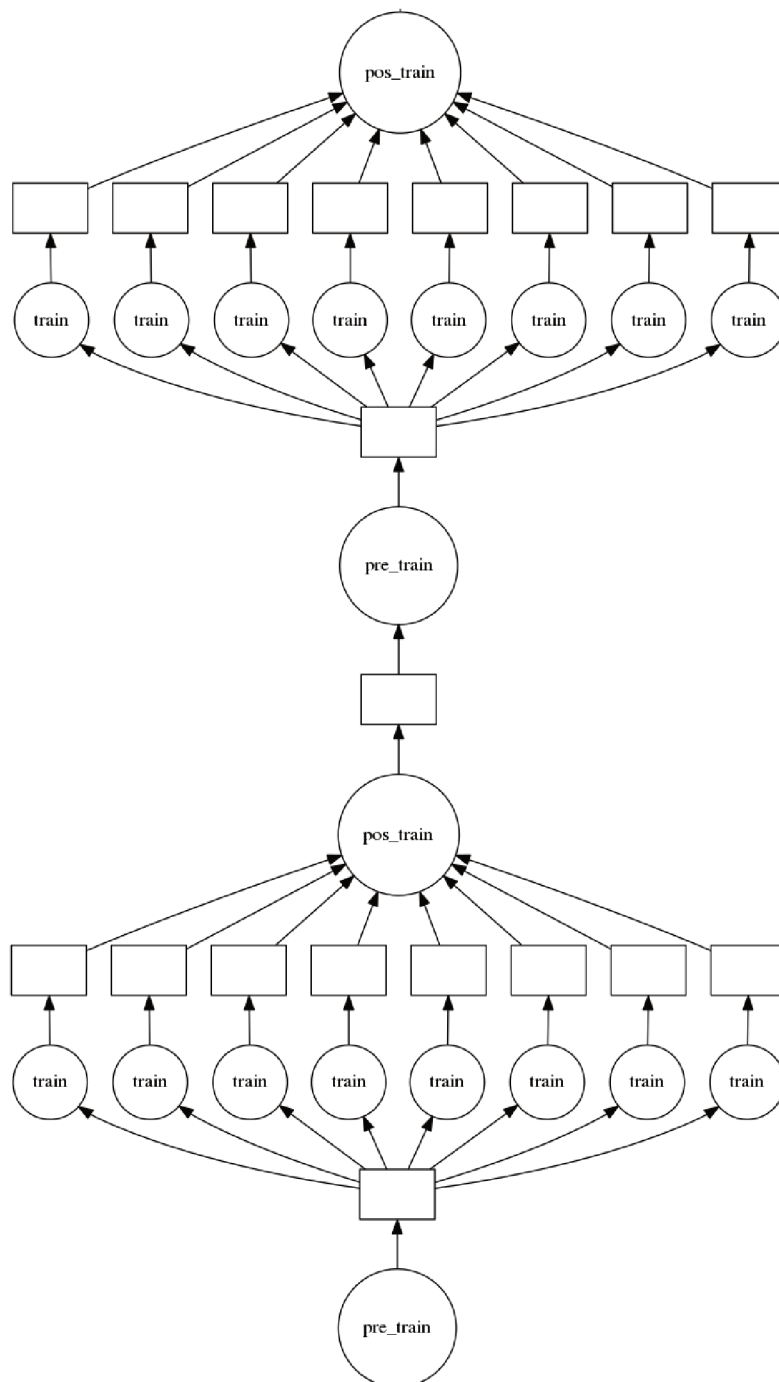


Figure 17 – Eight different models in Layer 0 and in Layer 1.

The first result that emerged quite clearly is the increase in diversity between the models. Table 5.10 shows the main measures of diversity. Any value in this table has a better diversity value when compared to table 5.8, for example. Just to mention a comparison, entropy changes from 0.01898 to 0.21620. A considerably better value.

Regarding the metrics obtained, table 5.11 shows the metrics obtained for the Layer 0 and table 5.12 shows the metrics for the Layer 1.

The best accuracy in the Layer 0 is obtained by the SVC algorithm (0.8703). At

Diversity measure	Min	Max	Average
Q statistics	0.65007	0.99395	0.86287
Correlation coefficient $\rho$	0.29211	0.88523	0.52019
Disagreement measure	0.04693	0.26040	0.17041
Double-fault measure	0.08967	0.26313	0.13373
	Value		
Entropy measure E	0.21620		
Kohavi-Wolpert variance	0.07455		

Table 5.10 – Diversity measurements between the models in the Layer 0.

Algorithm	Precision		Recall		F1-score		Accuracy
	Class 0	Class 1	Class 0	Class 1	Class 0	Class 1	
CatBoost	0.8317	0.8384	0.8402	0.8298	0.8359	0.8341	0.8350
LightGBM	0.7981	0.7899	0.7872	0.8007	0.7926	0.7953	0.7939
RF	0.7817	0.7750	0.7725	0.7840	0.7770	0.7795	0.7783
LR	0.7115	0.7143	0.7165	0.7092	0.7140	0.7117	0.7129
KNN	0.8229	0.8429	0.8478	0.8174	0.8352	0.8300	0.8326
SVC	0.8778	0.8630	0.8605	0.8801	0.8690	0.8715	0.8703
LSVC	0.7144	0.7134	0.7132	0.7147	0.7138	0.7141	0.7139
XGBoost	0.7279	0.6976	0.6764	0.7470	0.7012	0.7214	0.7117

Table 5.11 – Metrics for all models in the layer 0.

Algorithm	Precision		Recall		F1-score		Accuracy
	Class 0	Class 1	Class 0	Class 1	Class 0	Class 1	
CatBoost	0.8647	0.8762	0.8782	0.8625	0.8714	0.8693	0.8703
LightGBM	0.8798	0.8534	0.8483	0.8840	0.8638	0.8684	0.8661
RF	0.8686	0.8744	0.8755	0.8674	0.8720	0.8709	0.8715
LR	0.8641	0.8700	0.8711	0.8629	0.8676	0.8664	0.8670
KNN	0.8320	0.8468	0.8503	0.8282	0.8411	0.8374	0.8393
SVC	0.8778	0.8630	0.8605	0.8801	0.8690	0.8715	0.8703
LSVC	0.8624	0.8598	0.8594	0.8627	0.8609	0.8613	0.8611
XGBoost	0.8698	0.8752	0.8762	0.8687	0.8730	0.8719	0.8725

Table 5.12 – Metrics for all models in the layer 1.

the output of Layer 1, two models get the same precision, CatBoost and SVC itself, and two models improved this precision, Random Forest (0.8715) and XGBoost (0.8725). In addition, the vast majority of precision, recall and F1-score metrics show improvements in Layer 1 over Layer 0.

Unlike what happened with the first and second approaches of sections 5.2 and 5.3, these results are considered stable in the sense that they are consistent. Changes in the dataset settings or in the models hyperparameters continue to generate good results in the stacking output. In other words, it is not difficult to achieve good results with this approach.

After these conclusions, another experiment was made from this presented scenario. The best Layer 0 model has been removed, that is, the SVC model. Then, the complete stacking was retrained. As the Layer 0 models are still trained on the same dataset and are not dependent between each other, all results have been the same for the Layer 0, except for the absence of the removed SVC. As expected, CatBoost has become the most accurate model (0.8350). By the other side, training and metrics for all Layer 1 models have changed due to the absence of the SVC model (the input dataset of the Layer 1 has changed) and the results are shown in Table 5.13.

Algorithm	Precision		Recall		F1-score		Accuracy
	Class 0	Class 1	Class 0	Class 1	Class 0	Class 1	
CatBoost	0.8267	0.8334	0.8353	0.8247	0.8309	0.8290	0.8300
LightGBM	0.8384	0.8378	0.8378	0.8383	0.8381	0.8381	0.8381
RF	0.8262	0.8467	0.8515	0.8207	0.8387	0.8335	0.8361
LR	0.8167	0.8305	0.8343	0.8126	0.8254	0.8215	0.8235
KNN	0.8370	0.7561	0.7232	0.8590	0.7759	0.8043	0.7911
SVC	0.8224	0.8354	0.8389	0.8187	0.8306	0.8270	0.8288
LSVC	0.8145	0.8291	0.8331	0.8101	0.8237	0.8195	0.8216
XGBoost	0.8530	0.8277	0.8214	0.8583	0.8369	0.8427	0.8399

Table 5.13 – Metrics for all models in the layer 1 without SVC in the Layer 0.

Now, it is possible to notice that there are three models that are better than the best model of the Layer 0: LightGBM (0.8381), Random Forest (0.8361) and XGBoost (0.8399). In addition, the accuracy gain achieved by the Layer 1, in absolute terms, is greater than the scenario with the presence of the SVC. Another important finding is that the accuracy of the stacking as a whole has decreased due to the absence of SVC. This experience described here, among others carried out throughout the thesis, leads to these perceptions:

- In a scenario of high diversity between models, the less accurate the models of the first layers are, the easier it becomes to achieve significant advances in the last layers of the stacking.
- Once the first layer models tend to have better accuracy, the final stacking accuracy also improves, but it becomes more difficult to achieve great advances over the first layer models.
- The recipe for the stacking to keep improving even with the improvement of the first layer models is the continuous increase in diversity between the models.



## 5.5 FOURTH APPROACH: HIGH DIVERSITY ENSEMBLES

The next source of diversity to be studied is that coming from the dataset. Although the dataset is seen as a unitary data block, in many situations it is heterogeneous, that is, its columns can be grouped in different domains.

Just as an example, let's illustrate a practical situation like the insurance industry. In the same dataset, some columns may contain information about people, others about cars, others about the country's economy and so on.

Therefore, models could be trained not on the full dataset, but only on certain domains. The stacking would have the role of bringing all these groups together for the final prediction.

The parameters used to generate the dataset used by this experiment are depicted in the table 5.14.

Parameter	Value
Number of samples	50,000
Total number of features	60
Number of informative features	42
Number of redundant features	12
Number of repeated features	6
Number of clusters per class	4
Fraction of target assigned randomly	0.15
Class separation	0.9

Table 5.14 – Parameters used to generate the dataset.

The total number of features (60) is divided into three random groups of 20 features. For each group, a set of five models forms the first stacking layer, that is, Layer 0. For a fair comparison between these groups, the same models are used in the three groups. To compose the Layer 1 of the stacking, five models are used. Figure 18 illustrates this stacking.

The results achieved are quite positive. The diversity measures are very high and are shown in table 5.15. For the first time in this thesis, Q statistic had a negative value between two models (Q min), which means that they are making more errors in different samples than in the same (KUNCHEVA; WHITAKER, 2003). Almost all the values shown in the table 5.15 are the highest seen so far. Just to mention a few, the disagreement measure achieved 0.37396 on average. The entropy measure achieved 0.56690.

The metrics obtained are shown in table 5.16 (Layer 0) and table 5.17 (Layer 1). As expected due to the high diversity already shown, stacking provided a considerable gain in accuracy. The best Layer 0 model is LightGBM, present in group 3, with an accuracy of 0.6957. On the other hand, the best Layer 1 model, Random Forest, has

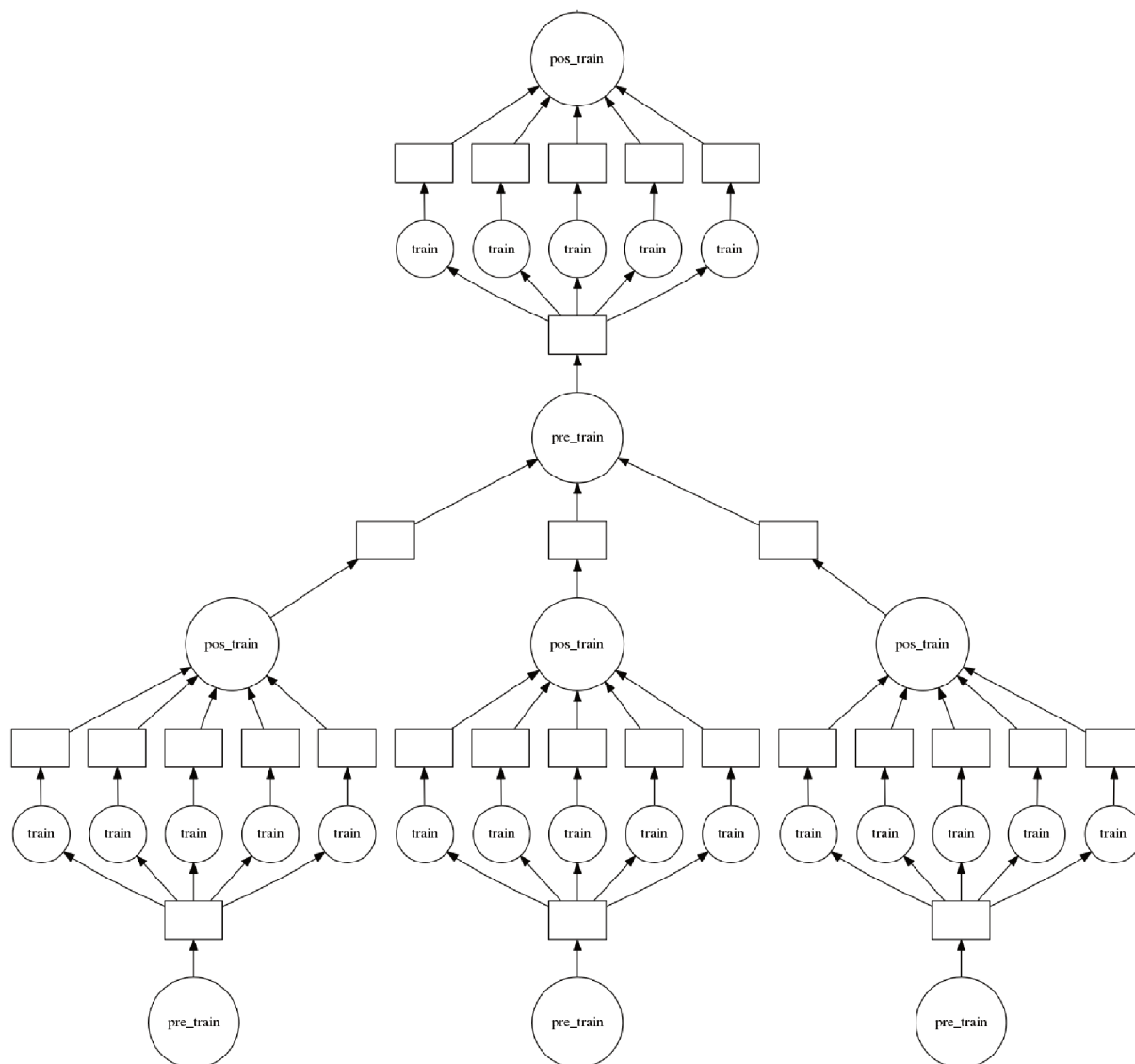


Figure 18 – Three groups of five models in the Layer 0 plus five models composing the Layer 1.

an accuracy of 0.7420.

It is important to mention that in this experience there are two sources of improvement brought by the stacking.

1. As each group is training only on 20 features and is not getting information about the remaining 40 features, it is expected that each group gathers information only about the features that it is training on. When stacking is done, all the dataset information is gathered and contributes to the gain shown. This type of gain did not occur in the experiments of the previous sections because in that case the models always trained on all features provided by the dataset. This process of "gathering" the information from these three groups could be done by other ensemble algorithms, such as, for example, a majority vote, but here stacking has

Diversity measure	Min	Max	Average
Q statistics	-0.11589	0.97002	0.34730
Correlation coefficient $\rho$	-0.05606	0.76204	0.18707
Disagreement measure	0.10447	0.51427	0.37396
Double-fault measure	0.12207	0.28413	0.16848
	Value		
Entropy measure E	0.56690		
Kohavi-Wolpert variance	0.17452		

Table 5.15 – Diversity measurements between all models in the Layer 0.

Algorithm	Precision		Recall		F1-score		Accuracy
	Class 0	Class 1	Class 0	Class 1	Class 0	Class 1	
Group 1							
LightGBM	0.6845	0.6982	0.7069	0.6754	0.6955	0.6866	0.6911
RF	0.6531	0.6804	0.7041	0.6275	0.6776	0.6529	0.6657
LR	0.6157	0.6137	0.6063	0.6230	0.6109	0.6183	0.6147
KNN	0.6622	0.6846	0.7025	0.6431	0.6818	0.6632	0.6727
LSVC	0.5679	0.5740	0.5852	0.5566	0.5764	0.5652	0.5709
Group 2							
LightGBM	0.6843	0.6786	0.6711	0.6917	0.6776	0.6851	0.6814
RF	0.6616	0.6618	0.6593	0.6641	0.6605	0.6630	0.6617
LR	0.6153	0.6170	0.6156	0.6166	0.6155	0.6168	0.6161
KNN	0.6354	0.6635	0.6923	0.6043	0.6626	0.6325	0.6482
LSVC	0.5553	0.5567	0.5538	0.5582	0.5545	0.5575	0.5560
Group 3							
LightGBM	0.6999	0.6916	0.6828	0.7084	0.6913	0.6999	0.6957
RF	0.6678	0.6625	0.6544	0.6758	0.6610	0.6691	0.6651
LR	0.6618	0.6548	0.6444	0.6720	0.6529	0.6633	0.6582
KNN	0.6513	0.6565	0.6593	0.6484	0.6553	0.6524	0.6539
LSVC	0.6184	0.6150	0.6053	0.6279	0.6118	0.6214	0.6167

Table 5.16 – Metrics for each group in the Layer 0.

been used.

2. The gain provided by the stacking itself in the sense shown in the previous sections of this chapter.

Continuing this experience, a new group of models are added in the Layer 0 of the stacking. However, in this case, the models of this group train on all 60 features of the dataset. The output data of this group composes the input dataset of the Layer 1 together with the three groups already described. This new stacking composition described here is shown in Figure 19. The goal with this is to evaluate the impact on the stacking of this “mix” between models that train in part of the dataset and others that train over the entire dataset.

Algorithm	Precision		Recall		F1-score		Accuracy
	Class 0	Class 1	Class 0	Class 1	Class 0	Class 1	
LightGBM	0.7107	0.7416	0.7576	0.6929	0.7334	0.7164	0.7252
RF	0.7414	0.7426	0.7418	0.7422	0.7416	0.7424	0.7420
LR	0.7200	0.7251	0.7264	0.7187	0.7232	0.7219	0.7225
KNN	0.7254	0.7352	0.7392	0.7212	0.7322	0.7282	0.7302
LSVC	0.7136	0.7189	0.7205	0.7120	0.7171	0.7155	0.7163

Table 5.17 – Metrics for all models in the layer 1.

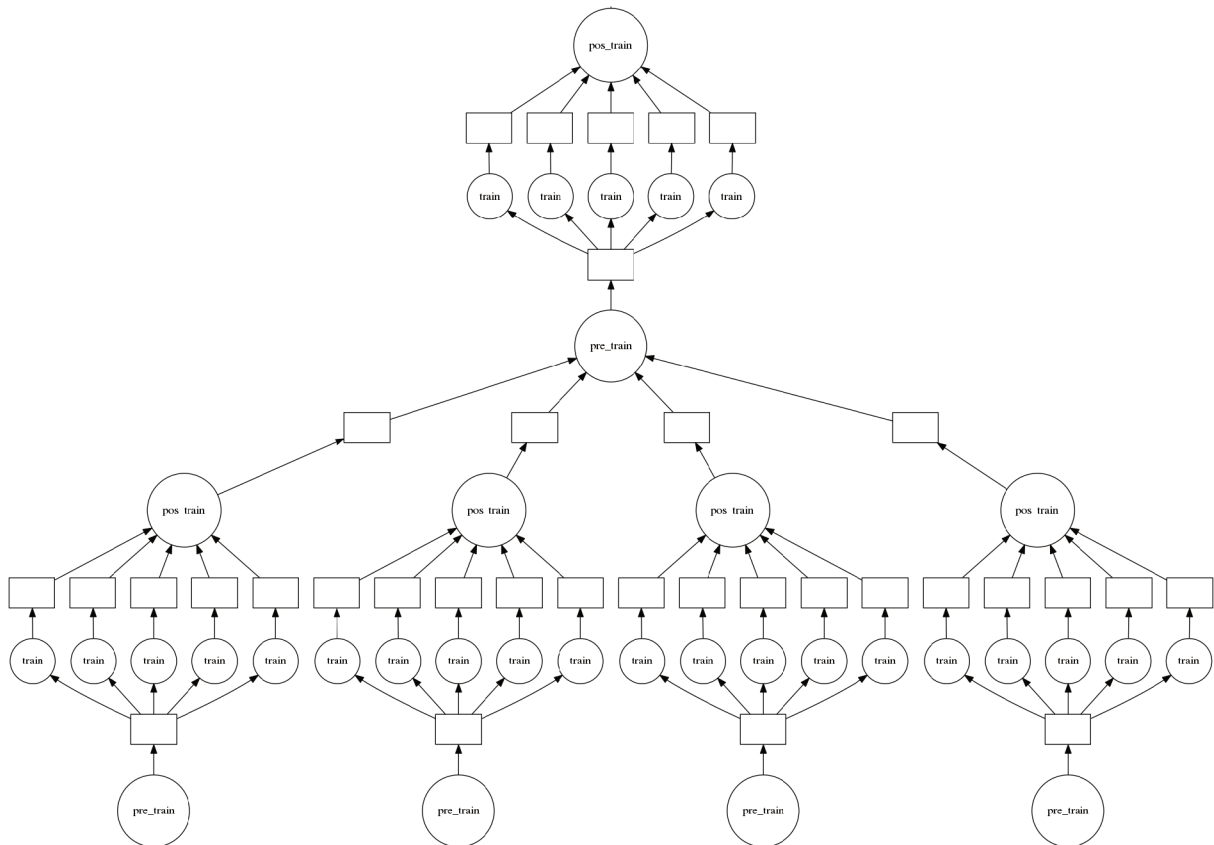


Figure 19 – Eight different models in Layer 0 and in Layer 1.

The metrics obtained for the fourth group inserted in the Layer 0 are shown in table 5.18. The metrics obtained for the Layer 1 in this new configuration are shown in table 5.19.

The best model in group 4, Random Forest, obtained an accuracy of 0.7881, while the best model in Layer 1, also Random Forest, obtained an accuracy of 0.7325. This suggests that the stacking could not be better than the Random Forest model running alone. However, it is possible to notice that some Layer 1 models have better accuracy than some models in group 4. This suggests that, in general, this result depends on several variables, which includes the hyperparameters of the models and, not least, the dataset. Many real datasets have specific characteristics that are not

Algorithm	Precision		Recall		F1-score		Accuracy
	Class 0	Class 1	Class 0	Class 1	Class 0	Class 1	
Group 4							
LightGBM	0.7793	0.7763	0.7738	0.7818	0.7766	0.7790	0.7778
RF	0.7902	0.7859	0.7832	0.7929	0.7867	0.7894	0.7881
LR	0.7036	0.6928	0.6822	0.7138	0.6927	0.7031	0.6980
KNN	0.7486	0.7772	0.7881	0.7364	0.7678	0.7563	0.7622
LSVC	0.6487	0.6616	0.6729	0.6370	0.6606	0.6491	0.6549

Table 5.18 – Metrics for the group 4 models contained in the Layer 0.

Algorithm	Precision		Recall		F1-score		Accuracy
	Class 0	Class 1	Class 0	Class 1	Class 0	Class 1	
LightGBM	0.6281	0.6436	0.6608	0.6102	0.6440	0.6265	0.6355
RF	0.7315	0.7335	0.7329	0.7320	0.7322	0.7327	0.7325
LR	0.3520	0.3609	0.3443	0.3687	0.3481	0.3647	0.3565
KNN	0.7144	0.7187	0.7197	0.7134	0.7170	0.7160	0.7165
LSVC	0.3043	0.3177	0.2958	0.3265	0.3000	0.3220	0.3112

Table 5.19 – Metrics for all models in the Layer 1.

being considered in this artificial dataset. Therefore, due to these untested scenarios, it is still possible that this type of stacking brings good results.

## 5.6 PARALLEL VS SEQUENTIAL COMPUTING PERFORMANCE DIFFERENCES

One of the objectives of this thesis is to compare the parallel approach developed in the architecture built for this work against a sequential approach in terms of training time. This section reports the results obtained.

For this experiment, the same stacking reported in the 5.3 section has been used. Layer 0 is composed of 162 models plus 1 model in Layer 1. The accuracy of the models are not observed here, only the training times.

Three different algorithms have been tested: LightGBM, XGBoost and CatBoost. As these algorithms also run the training in parallel, and by default use all available cores, it was necessary to limit the number of threads to a single one. Otherwise, the results obtained here could not be attributed to the developed architecture because the parallelism would also be coming from the algorithms. The goal here is to evaluate only the architecture.

Table 5.20 illustrates the times obtained after training the stacking. The computer used has eight cores with the Ubuntu 18.04 LTS operating system. It is observed that the parallel execution is better by a factor of almost five times. It is important to highlight here that it is not possible and neither is the objective to compare the performance between the shown algorithms because they are being executed with different hyperparameters.

The objective is to compare the “Sequential” column against the “Parallel” column for each algorithm.

Algorithm	Sequential	Parallel
LightGBM	11min 19s	2min 11s
XGBoost	10min 13s	2min 04s
CatBoost	54min 32s	10min 55s

Table 5.20 – Training user times running sequentially and in parallel.

This experience alone would be sufficient to validate the parallelism proposed by the developed architecture. However, only as an investigation, another experience was performed. The parallelism coming from the algorithms and from the architecture are used together. Table 5.21 illustrates this experiment carried out with LightGBM. The training was performed several times, each time with a different number of threads (LightGBM allows to configure the number of threads to be used by it). In this table, “user time” means the CPU time actually consumed by the code being executed, “system time” means the time spent outside the code through system calls. These two times (user and system) are the sum of all the core times. That is why this time is longer than the wall time when more than one thread is used. “Wall time” is simply the total elapsed time.

Threads	Sequential times			Parallel times		
	User	System	Wall	User	System	Wall
1	11min 19s	2.84 s	11min 22s	17min 31s	3.3 s	2min 39s
2	12min 48s	3.76 s	6min 26s	24min 30s	8min 30s	4min 17s
3	14min 23s	5.33 s	4min 49s	24min 5s	10min 6	4min 21s
4	15min 42s	6.11 s	3min 57s	25min 47s	13min 26s	4min 58s
5	23min 34s	11.4 s	4min 45s	27min 27s	16min 42s	5min 34s
6	25min 57s	12.5 s	4min 21s	29min 16s	20min 42s	6min 16s
7	27min 17s	12.4 s	3min 55s	30min 32s	23min 26s	6min 46s
8	32min 14s	26min 29s	7min 21s	31min 51s	25min 56s	7min 13s

Table 5.21 – Sequential and parallel times when running the algorithms with more than one thread.

Looking at table 5.21, there are two different scenarios. In sequential execution, training times decrease in general as the number of threads increases. However, in parallel execution, training times increase as the number of threads increases. This thesis does not go deeply into this behavior. Only as a hypothesis, it is likely that the threads are blocking each other due to the sharing of some resource. Note that the system times increase as well. The section 6.1 on future work has a suggestion to improve this scenario using other schedulers offered by Dask.

Finally, an interesting observation. When parallelism comes only from the architecture, the wall time obtained (2min 39s) is less than the parallelism that comes only from the algorithm (7min 21s). This comparison can be seen in table 5.21 between the parallel approach with one thread versus the synchronous approach with eight threads. Even the best sequential time (3min 55s with seven threads) remains worse.

## 6 CONCLUSIONS

This work presented a very actual, broad and complex objective: study the ensemble in order to show a set of approaches that are likely to achieve a good result, that is, to have a better result than any of the individual models that make up the ensemble.

To achieve such goal, it was essential to develop an architecture that would allow the rapid elaboration of experiments with efficient execution. The development of this architecture, in turn, brought a series of challenges. First, it required a survey of the main tools available in the machine learning community. This research was fundamental because the algorithms used in this thesis are provided by different research groups and, due to the nature of the stacking, these algorithms would need to be integrated. Second, the development of the architecture was challenging: parallelism, concurrent access to datasets and definition of the correct training sequence were among the main obstacles encountered. At the end of the work, it can be concluded that the developed architecture was sufficient for the desired experiments. In addition, even more complex forms of stacking could have been done, with several Layers interconnected in more sophisticated ways.

Regarding the results obtained with the study of the ensemble, the work started trying to prove the results already mentioned in the literature. This proof was obtained in practice, starting with the stacking of similar models and with little diversity. The gains made with the stacking were either small or did not exist. The experiences developed after this evolved in the sense of always increasing the diversity between the models to seek greater gains with the stacking. At the end, this relationship between increased diversity and improvement brought by stacking could be observed. The first clear observation of this came in the section 5.4 when several different models were used in the stacking. The conclusions of this work on stacking performance are shown in the following items:

- To obtain accuracy improvements with the stacking, it is very important to use models of different natures, mainly in the first Layers.
- The diversity measures are good indicators of the stacking capacity to improve the final result.
- Models with high diversity and low accuracy (weak models) make it easier to gain more from the stacking.
- High accuracy models improve the final stacking accuracy, but at the same time make the improvement brought by the stacking more difficult. The introduction of models with high accuracy in the stacking should always be followed by the introduction of diversity so that the stacking continues to bring good results.



- The way the dataset is used in the stacking can also bring diversity to it. Studying the dataset and its features can be a good option before starting the modeling. Sometimes, some models do not need to train on all features.
- There is no ready recipe on how stacking should be done. There are many variables in question that can change the results.

Regarding the performance of the parallel architecture proposed, the experiments show an almost five times gain in an eight cores processor. The greater the number of processor cores are used, the greater this difference becomes.

Finally, many of the ensemble properties mentioned by the literature were put into practice and shown in this thesis. The developed platform opened up several possibilities for experiments. Some of them have been shown in this thesis for the conclusions here presented. Others are promising experiments to be developed and are explained in the next section 6.1.

## 6.1 FUTURE WORK

This thesis has a very broad scope and there are numerous opportunities for improvements and future work. This chapter presents the future work in order of priority, that is, improvements that would bring the best results given the current state of this work.

- Dataset

The dataset is a great source of diversity and this work uses only datasets generated through software and with pre-defined parameters for the reasons explained in 5.1. This approach has scientific advantages for a fair comparison between different approaches. Real datasets from society do not have some of these advantages, but they do have several sources of diversity to be explored. Many different distributions might be contained in the same dataset. The unknown function  $y = f(x)$  shown in chapter 2 can be more complex for data coming from real datasets than the function that generates the datasets used in this thesis.

- Feature engineering

The datasets used in this thesis are totally numeric, but the data coming from real datasets might have other non-numeric types, such as categorical, for example, and the way the feature engineering of these data is done can have a great impact on training. In other words, feature engineering is a source of diversity to be explored. Different approaches can be studied and correlated with their consequence for the final stacking accuracy.

- Study of the best model combinations

When constructing the stacking, there are several options of different models to be added under different conditions. A methodology for this could be developed. The work presented by (CARUANA et al., 2004) is a good reference in this area. In other words, a study on which models often have the greatest diversity among themselves and in which situations they should be used. Despite being a complex and wide area, this is a very frequent question when constructing the stacking.

- Interpretability study

Interpretability is a widely discussed topic in machine learning. There is no doubt that stacking makes the interpretability of the model as a whole more complex. It would be interesting to bring the discussions present in this area and also some solutions for the presented architecture.

- Neural networks

Neural networks are being used widely by the machine learning community and, in many cases, are generating quite high accuracy. However, neural networks were not an algorithm studied in this thesis. It would be interesting to observe the performance of the stacking by also inserting neural networks models.

- Include new problem classes

The binary classification problems studied in this thesis are quite common in machine learning, but there are other problems to be addressed. Multiclass classification and regression problems are also challenging and may open up new research fronts. It would be a good approach to study stacking with different classes of problems, such as classification and regression, and the diversity brought by this.

- Add new schedulers

Dask offers at least four types of schedulers: threads, processes, synchronous and distributed. The current architecture supports thread and synchronous schedulers. The other schedulers could be implemented to open up new possibilities and perhaps accelerate even more the tasks execution time.

## BIBLIOGRAPHY

BOLAND, Philip J. Majority system and the Condorcet jury theorem. **Statistician**, v. 38, n. 3, p. 181–189, 1989.

BREIMAN, Leo. Bagging Predictors. **Machine Learning**, p. 123–140, 1996. DOI: 10.1023/A:101805431.

BREIMAN, Leo. Random Forests. **Machine Learning**, p. 5–32, 2001. DOI: 10.1023/A:1010933404324.

BROWN, Gavin; KUNCHEVA, Ludmila I. “Good” and “Bad” Diversity in Majority Vote Ensembles. In: EL GAYAR, Neamat; KITTNER, Josef; ROLI, Fabio (Eds.). **Multiple Classifier Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. P. 124–133.

BROWN, Gavin; WYATT, Jeremy, et al. Diversity Creation Methods: A Survey And Categorisation. **Information Fusion**, v. 6, p. 5–20, Mar. 2005. DOI: 10.1016/j.inffus.2004.04.004.

CARUANA, Rich et al. Ensemble Selection from Libraries of Models. In: PROCEEDINGS of the Twenty-First International Conference on Machine Learning. Banff, Alberta, Canada: Association for Computing Machinery, 2004. (ICML '04), p. 18. DOI: 10.1145/1015330.1015432. Available from: <https://doi.org/10.1145/1015330.1015432>.

CHEN, Tianqi; GUESTRIN, Carlos. XGBoost: A Scalable Tree Boosting System. In: PROCEEDINGS of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco, California, USA: ACM, 2016. (KDD '16), p. 785–794. DOI: 10.1145/2939672.2939785. Available from: <http://doi.acm.org/10.1145/2939672.2939785>.

FAWCETT, Tom. An Introduction to ROC Analysis. **Pattern Recogn. Lett.**, Elsevier Science Inc., USA, v. 27, n. 8, p. 861–874, June 2006. ISSN 0167-8655. DOI: 10.1016/j.patrec.2005.10.010. Available from: <https://doi.org/10.1016/j.patrec.2005.10.010>.

FREUND, Yoav; SCHAPIRE, Robert E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. In: 1, p. 119–139. DOI: 10.1006/jcss.1997.1504. Available from: <http://dx.doi.org/10.1006/jcss.1997.1504>.

FRIEDMAN, Jerome H. Greedy Function Approximation: A Gradient Boosting Machine. **Annals of Statistics**, v. 29, p. 1189–1232, 2001.

HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome. **The Elements of Statistical Learning**: Data Mining, Inference, and Prediction. 2. ed. Stanford, California: Springer, 2008.

HE, Xiaogang et al. Spatial downscaling of precipitation using adaptable random forests. **Water Resources Research**, v. 52, 2016. DOI: 10.1002/2016WR019034.

JORDAN, M. I.; MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. **Science**, American Association for the Advancement of Science, v. 349, n. 6245, p. 255–260, 2015. ISSN 0036-8075. DOI: 10.1126/science.aaa8415. eprint: <https://science.sciencemag.org/content/349/6245/255.full.pdf>. Available from: <https://science.sciencemag.org/content/349/6245/255>.

KE, Guolin et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Ed. by I. Guyon. Curran Associates, Inc., p. 3146–3154, 2017. Available from: <http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf>.

KEARNS, Michael. Thoughts on Hypothesis Boosting. **Unpublished manuscript (Machine Learning class project)**, Dec. 1988.

KEARNS, Michael; VALIANT, Leslie G. Cryptographic Limitations on Learning Boolean Formulae and Finite Automata. In: PROCEEDINGS of the Twenty-first Annual ACM Symposium on Theory of Computing. Seattle, Washington, USA: ACM, 1989. (STOC '89), p. 433–444. DOI: 10.1145/73007.73049. Available from: <http://doi.acm.org/10.1145/73007.73049>.

KUNCHEVA, Ludmila I.; WHITAKER, Christopher J. Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy. **Machine Learning**, v. 51, n. 2, p. 181–207, May 2003. ISSN 1573-0565. DOI: 10.1023/A:1022859003006. Available from: <https://doi.org/10.1023/A:1022859003006>.

MITCHELL, Thomas M. **Machine Learning**. 1. ed. USA: McGraw-Hill, Inc., 1997. ISBN 0070428077.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

POLIKAR, Robi. Ensemble based systems in decision making. **IEEE Circuits and Systems Magazine**, v. 6, n. 3, p. 21–45, Third 2006. ISSN 1531-636X. DOI: 10.1109/MCAS.2006.1688199.

RIDGEWAY, Greg. Generalized Boosted Models: A guide to the gbm package, Aug. 2007.

RUSSELL, Stuart; NORVIG, Peter. **Artificial Intelligence: A Modern Approach**. 3rd. USA: Prentice Hall Press, 2009. ISBN 0136042597.

SCHAPIRE, Robert E. The strength of weak learnability. **Machine Learning**, p. 197–227, 1990. DOI: 10.1007/BF00116037.

TANG, E. K.; SUGANTHAN, P. N.; YAO, X. An Analysis of Diversity Measures. **Mach. Learn.**, Kluwer Academic Publishers, USA, v. 65, n. 1, p. 247–271, Oct. 2006. ISSN 0885-6125. DOI: 10.1007/s10994-006-9449-2. Available from: <https://doi.org/10.1007/s10994-006-9449-2>.

VEEN, Hendrik Jacob van; DAT, Le Nguyen The; SEGNINI, Armando. **Kaggle Ensembling Guide**. 2015. Available from: <https://mlwave.com/kaggle-ensembling-guide/>. Visited on: 11 Jan. 2020.

WOLPERT, David H. Stacked Generalization. **Neural Networks**, v. 5, p. 241–259, 1992.

## APPENDIX A – ALTERNATIVE STACKING CONFIGURATION TO THE SECOND APPROACH

When presenting the results in chapter 5, section 5.3 mentioned that this thesis found a stacking configuration for that scenario where the final stacking result was better than the best Layer 0 model. This configuration is presented here.

Table A.1 shows the parameters used to generate the dataset. Table A.2 shows the hyperparameters used. There are a total of 9 hyperparameter configurations (three learning rates times three max tree depth), but each of these configurations is replicated 18 times so that the generated dataset has 162 columns. This result was found randomly due to a software bug, but turned out to be a surprising result. Table A.3 shows the metrics for all models. The final stacking accuracy is 0.9705 which is better than the best Layer 0 model, 0.9681.

Parameter	Value
Number of samples	100,000
Total number of features	20
Number of informative features	14
Number of redundant features	4
Number of repeated features	2
Fraction of target assigned randomly	0.015

Table A.1 – Parameters used to generate the dataset.

Booster parameter	Layer 0 models	Model (1, 0, 0)
Learning rate	0.15, 0.25, 0.35	0.2
Max tree depth	4, 5, 6	10
Metric	AUC	AUC

Table A.2 – Booster parameters used by the stacking models.

Metric	Layer 0 - Min	Layer 0 - Max	Layer 0 - Mean	Model (1, 0, 0)
Precision - class 0	0.930390	0.964955	0.953750	0.9685
Precision - class 1	0.932086	0.971309	0.958876	0.9725
Recall - class 0	0.932678	0.971689	0.959357	0.9728
Recall - class 1	0.929780	0.964489	0.953202	0.9682
F1-score - class 0	0.931532	0.968310	0.956545	0.9707
F1-score - class 1	0.930932	0.967887	0.956030	0.9703
Accuracy	0.931233	0.968100	0.956289	0.9705

Table A.3 – Metrics for all models in layer 0 and the model in the Layer 1 (1, 0, 0).