Pedro José Silva Pereira
Modern Optimization of Predictive Models:
An Application to Mobile Performance Marketing
UMinho | 2021

Universidade do Minho
Escola de Engenharia

Pedro José Silva Pereira

Modern Optimization of Predictive Models: An Application to Mobile Performance Marketing

December, 2021

Universidade do Minho
Escola de Engenharia
Departamento de Sistemas de Informação

Pedro José Silva Pereira

Modern Optimization of Predictive Models:

An Application to Mobile Performance Marketing

PhD Thesis
Doctoral Program on Information Systems and Technology

Supervisors:
Professor Doctor Paulo Alexandre Ribeiro Cortez
Professor Doctor Rui Manuel Ribeiro de Castro Mendes

December 2021

ACKNOWLEDGEMENTS

This PhD journey was undoubtedly a lifetime challenge and I am fortunate to have been surrounded with incredible people along the way, who taught me so much, and to whom I would like to dedicate this work.

In the first place, I want to thank to both my supervisors, Professor Paulo Cortez and Professor Rui Mendes, for the knowledge and experiences shared, for their availability to help, their enormous patience and for the trust they placed in me. Whenever difficulties emerged, they always had the right words and advice to motivate and inspire me. It has been a great honour to work with such examples of professionalism, commitment, ethics and rigor. I would like to give a special thanks to Professor Cortez, who supervised my MSc thesis and with whom I have been working ever since, for the research and teaching opportunities provided. He was the responsible for awakening my interest in these areas and I am truly grateful for that.

To my friends, the older ones and the ones I met during this journey, thank you for the comradeship, to help me keeping focused, the shared knowledge and professional experiences, for being my source of confidence and motivation, and for always being present, even when being physically distant. A heartfelt thank you, I am grateful for sharing this adventure with you! I also would like to thank to my parents and brother, who always supported and encouraged me during this PhD project, providing everything they could to watch me succeed.

Finally, I owe a few appreciation words to my Machine Learning research colleagues, with whom I have been working during the last two years, for the great team working environment and for their contribution to my professional growth.

STATEMENT OF INTEGRITY

ABSTRACT

The increase of mobile devices usage has leveraged digital business opportunities, particularly in the advertising sector. In particular, the Mobile Performance Marketing (MPM) industry has been a target of vast investments in the last years. This industry deals with advertising campaigns (owned by advertisers) presented on digital spaces (owned by publishers), such as mobile applications. In short, users are redirected to campaign pages in order to access publishers content. Between advertisers and publishers, intermediary companies perform matching between users and campaigns, by means of a Demand-Side Platform (DSP), aiming to lead the users to a purchase. However, the currently used mechanisms to perform this matching are rather rigid, using eventual profit or simple statistical rules to select the campaign to be displayed, instead of using other data attribute relationships. Overall, this DSP assignment method tends to produce a tiny conversion rate (e.g., 1%) between user redirects and purchases. Thus, improving the DSP performance with the application of a Machine Learning (ML) approach could potentially improve this business.

ML has proved to be extremely important in several domains, by being able to extract complex patterns from data and providing useful insights to decision makers. During this PhD, we particularly address the multi-objective Modern Optimization of predictive models, a much less researched topic when compared with traditional ML approaches. In particular, we use Evolutionary Algorithms (EAs) to design and evolve ML models, using MPM as the final (and target) use case. At an initial stage of this PhD thesis, when we did not had access to MPM data, we addressed a distinct ML use case. In effect, we proposed several neuroevolution models for multi-step ahead time series Prediction Intervals (PIs), based on a Pareto-based multi-objective EA that simultaneously considered PI coverage and width. This research included the proposal of a novel and robust evaluation method for multi-objective ML models, which vertically aggregates similar solutions using the Wilcoxon median and 95% confidence intervals. At a later stage, when MPM data was available, we developed two innovative approaches for the design and creation of Evolutionary Decision Trees (EDTs), using a multi-objective implementation of Grammatical Evolution (GE) that simultaneously considered EDTs predictive performance and complexity. Using a realistic and robust experimentation, with real-world data from the OLAmobile company, we have shown that the proposed GE is capable of evolving valuable prediction models for the MPM domain.

Keywords: Multi-Objective Machine Learning, Evolutionary Algorithms, Mobile Performance Marketing

RESUMO

O aumento na utilização de dispositivos móveis alavancou oportunidades de negócios digitais, particularmente no setor da publicidade. Em particular, a indústria de *Mobile Performance Marketing* (MPM) tem sido alvo de grandes investimentos nos últimos anos. Esta indústria lida com campanhas publicitárias de *advertisers*, apresentadas em espaços digitais de *publishers*, como aplicações móveis. Em suma, os utilizadores são redirecionados para páginas de campanhas de modo a poderem aceder ao conteúdo dos *publishers*. Entre *advertisers* e *publishers*, empresas intermediárias direcionam utilizadores para campanhas, utilizando *Demand-Side Platforms* (DSPs), com o intuito de os levar a realizar uma compra. Contudo, os mecanismos utilizados atualmente para realizar este direcionamento são bastante rígidos, utilizando o eventual lucro ou regras de estatística simples para selecionar a campanha a apresentar, ao invés de utilizarem outras relações entre os atributos de dados. No geral, o método atualmente utilizado pelo DSP tende a produzir uma reduzida taxa de conversão entre utilizadores redirecionados e compras efetuadas (e.g., 1%), que poderia ser melhorada com a aplicação de uma abordagem de *Machine Learning* (ML).

O ML já provou ser extremamente importante em diversos domínios, sendo capaz de extrair padrões complexos dos dados e fornecer informação útil para a tomada de decisão. Durante este doutoramento, abordamos a Otimização Moderna e multiobjetivo de modelos preditivos, um tópico ainda pouco investigado. Em particular, utilizamos Algoritmos Evolucionários (AEs) para criar modelos de ML, utilizando o MPM como principal caso de estudo. Inicialmente, quando ainda não tínhamos acesso a dados de MPM, abordamos um caso de estudo de ML distinto. Na realidade, propusemos diferentes modelos Neuroevolucionários para previsão de intervalos (PIs) em séries temporais, recorrendo a um AE multiobjetivo que simultaneamente considerava o erro e a largura dos mesmos. Esta investigação incluiu a proposta de um método robusto para a avaliação de modelos multiobjetivo de ML, que agrega soluções usando a mediana de Wilcoxon e intervalos de 95% de confiança. Posteriormente, quando os dados de MPM estavam disponíveis, desenvolvemos duas abordagens inovadoras para a criação de Árvores de Decisão, utilizando um AE multiobjetivo que considerava simultaneamente o desempenho preditivo e a complexidade destas. Com uma experimentação robusta e realista, mostramos que as abordagens propostas são capazes de evoluir modelos preditivos valiosos para este domínio.

Keywords: Machine Learning Multiobjetivo, Algoritmos Evolucionários, Mobile Performance Marketing

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ACO Ant Colony Optimization.

AI Artificial Intelligence.

ANN Artificial Neural Network.

API Application Programming Interface.

AUC Area Under the Curve.

BNF Backus–Naur Form.

cgNEAT content-generating NeuroEvolution of Augmenting Topologies.

CPA Cost Per Acquisition.

CPC Cost Per Click.

CPI Cost Per Install.

CPM Cost Per Mille.

CPV Cost Per View.

CRISP-DM CRoss-Industry Standard Process for Data Mining.

CTR Click Through Rate.

CVR Conversion Rate.

CWC Coverage Width-based Criterion.

DE Differential Evolution.

DL Deep Learning.

DM Data Mining.

DSP Demand-Side Platform.

DSRM-IS Design Science Research Methodology for Information Systems.

DT Decision Tree.

EA Evolutionary Algorithm.

EANN Evolutionary Artificial Neural Network.

EC Evolutionary Computation.

EDA Estimation of Distribution Algorithm.

EDT Evolutionary Decision Tree.

FNN Feedforward Neural Network.

GA Genetic Algorithm.

GE Grammatical Evolution.

GEP Gene Expression Programming.

GNARL GeNeralized Acquisition of Re-current Link.

GP Genetic Programming.

HyperNEAT Hypercube-based NeuroEvolution of Augmenting Topologies.

ICT Information and Communication Technology.

IDF Inverse Document Frequency.

IS Information System.

IT Information Technologies.

JSON JavaScript Object Notation.

KDD Knowledge Discovery from Databases.

LE Lamarckian Evolution.

LR Logistic Regression.

LTV Lifetime Value.

LUBE Lower and Upper Bound Estimation.

LUBEX Lower and Upper Bound Estimation eXtension.

MGEDT Multi-objective Grammatical Evolution Decision Tree.

MGEDTL Multi-objective Grammatical Evolution Decision Tree Lamarckian evolved.

ML Machine Learning.

MLP Multilayer Perceptron.

MLUBE Multi-objective evolutionary algorithm Lower and Upper Bound Estimation.

MMNEAT Modular Multi-objective NeuroEvolution of Augmenting Topologies.

MO Multi-Objective Optimization.

MOEA Multi-Objective Evolutionary Algorithm.

MSE Mean Squared Error.

NB Naive Bayes.

NEAT NeuroEvolution of Augmenting Topologies.

NMPIW Normalized Mean Prediction Interval Width.

NN Neural Network.

NSGA-II Non-dominated Sorting Genetic Algorithm II.

PCP Percentage Categorical Pruning.

PI Prediction Interval.

PICE Prediction Interval Coverage Error.

PICP Prediction Interval Coverage Probability.

PMBGA Probabilistic Model-Building Genetic Algorithm.

PROMOS PRediction and Optimization of MObile Subscription marketing campaigns.

PSO Particle Swarm Optimization.

R&D Research and Development.

RBF Radial Basis Function.

RF Random Forest.

RNN Recurrent Neural Network.

ROC Receiver Operating Characteristic.

rtNEAT real-time NeuroEvolution of Augmenting Topologies.

RW Rolling Window.

SEEA Simple Elitist Evolutionary Algorithm.

SLP Single Layer Perceptron.

SMOTE Synthetic Minority Oversampling Technique.

SOM Self-Organized Map.

SPEA 2 Strength Pareto Evolutionary Algorithm 2.

SVM Support Vector Machine.

TSF Time-Series Forecasting.

TWEANN Topology and Weight Evolving Artificial Neural Network.

XAI Explainable Artificial Intelligence.

XB XGboost.

Part I

INTRODUCTION AND BACKGROUND

1

INTRODUCTION

1.1 Motivation

Advances in *Information Technologies (IT)* led to an exponential increase on the amount of produced and stored data. In particular, the growth on smart devices usage (e.g., tablets, smartphones) contributed to this data growth and also leveraged new business opportunities for digital markets, including advertising. Particularly, a market termed Performance Marketing, or Performance-based Advertising, has been a target of attention and investments in the last few years (Statista, 2021). In this market, advertising campaigns are displayed to users, intending to lead them to purchase a product or subscribe a service (resulting in a conversion). This PhD thesis presents an application of modern optimization of predictive algorithms to the mobile sector of Performance Marketing, collaborating with a company that operates in this sector (OLAmobile). Their business, illustrated in Figure 1, comprehends 4 main stakeholders: advertisers, publishers, users and intermediary entities, such as the company under study. Advertisers are entities with ad campaigns regarding their products or services and that need popular digital spaces (e.g., news websites, online mobile games) to display their campaigns. Publishers own these digital spaces and attract a vast audience of users to their content. *Demand-Side Platforms (DSPs)*, owned by intermediaries, are platforms responsible for matching users to advertisements, aiming to lead the to a conversion (purchase or subscription) and for assuring the respective monetization flows (e.g., for the publishers).

OLAmobile operates in the Mobile Performance Marketing industry since 2011, being responsible for redirecting quality users to their partners/advertisers mobile services campaigns. In this market, compensation only occurs if mensurable metrics are presented, which means that this company is paid for each succeeded purchase or service subscription, in what is termed as *Cost Per Acquisition (CPA)* or *Cost Per Install (CPI)*. Campaigns are displayed to users through a smart link, and each time they click on it, a new event called redirect is generated and stored. This event consists in redirecting the user to the campaign page, where a different event can be performed: a sale. There are different types of sales, namely product purchases (e.g., mobile games, musics) or service subscriptions (e.g., news, ringing sounds), with the latter being of greater interest to advertisers, since users perform continuous payments in order to maintain the subscription. Advertisers hire intermediary companies based on

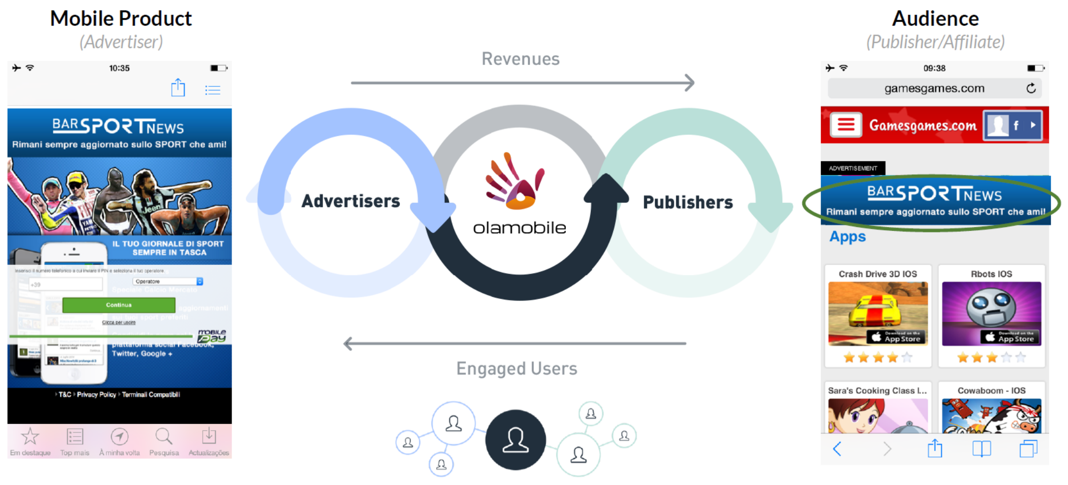


Figure 1: Illustration of the OLAmobile business.

their DSP performance, often using: a *Click Through Rate (CTR)* measure, which corresponds to the number of user clicks on campaigns divided by the number of times these were presented; and a user *Conversion Rate (CVR)*, which measures the probability that a particular user will perform a conversion after clicking an ad link. Thus, a good DSP performance in terms of both CTR and CVR increases the chances of conversions and, consequently, increases profits for 3 of the main stakeholders (advertisers, publishers and intermediaries).

In the particular case of the OLAmobile company, several million redirects are generated hourly, from which only a small percentage (around 1%) originates a conversion and there is a limited set of information that can be used, due to technological and privacy limitations. Moreover, according to OLAmobile specialists, neither the company nor its main direct and indirect competitors use *Machine Learning (ML)* techniques in their DSP to match a campaign to a given user, nor do they present the more relevant product to the users. Instead, mobile campaign selection is performed based on eventual profit in case of a conversion, which benefits the exhibition of higher profitable campaigns. Hence, improving the DSP process of matching users to campaigns of their interest would have a direct impact on this market and may yield a noticeable competitive edge to this industry. In this doctoral thesis, we address this particular Mobile Performance Marketing issue, using OLAmobile data, by optimizing Predictive ML models using Modern Optimization approaches.

In the past, several studies adopted Modern Optimization to design predictive models, outperforming traditional ones (Stanley and Miikkulainen, 2002; Mendes et al., 2002; Cortez and Donate, 2014; Donate and Cortez, 2014; Ojha et al., 2017; Pereira et al., 2017). However, most of the proposed methods on these studies are evaluated in static environments with smaller amounts of data, when compared with the Mobile Performance Marketing industry. In effect, this industry deals with high-speed Big Data, with several campaigns often emerging and disappearing, and the DSP has a limited time to provide a correct campaign, specifically 10 milliseconds. Thus, during this PhD, we intend to focus on the research and development of an advanced ML system, involving learning algorithms (e.g., *Artificial Neural Networks (ANNs)*, *Decision Trees (DTs)*), trained using relevant state-of-the-art modern optimization methods (e.g., *Evolutionary Algorithms (EAs)*). The goal is to optimize, using *Evolutionary Computation (EC)*, predictive models that allow to perform the optimal matching between the advertised product and the interested and relevant audience to that product. In particular, we focus on multi-objective implementations, aiming to deal with two or more measures that may fit the company requirements. Moreover, although EAs tend to be reasonable in terms of computational cost (Michalewicz et al., 2006), these methods are slower when compared with traditional ML algorithms and the mobile marketing industry has tight time limitations. Therefore, we intend to explore parallelism approaches to address this issue and fulfill the company's requirements.

1.2 Objectives

This doctoral thesis was partially developed under a *Research and Development (R&D)* project named *PRediction and Optimization of MObile Subscription marketing campaigns (PROMOS)*[1], related with the use of *Artificial Intelligence (AI)* and Modern Optimization applied to the Mobile Performance Marketing industry, born from a consortium between OLAmobile company and University of Minho. Before this project, companies operating in this sector, not only OLAmobile but also its partners and competitors, resort to simple mathematical formulas based on predefined variables to redirect their users to the most appropriate product campaign. In particular, the used data attributes only consider user information (e.g., country, operative system) and the campaign selection is based on the eventual obtained profit in case of a purchase.

With this approach, campaigns regarding higher profitable products have a higher probability of being displayed, even though it may not be the more interesting option for the end user. Moreover, this method is currently translated in a success CVR rate of nearly 1% between redirects and purchases. Since compensation only occurs when a user performs a purchase, augmenting this rate would directly increase profit for companies as OLAmobile. Hence, developing a more accurate method would have a direct influence on business, not only for the company under study, but also for other intermediary companies, advertisers and publishers. According to OLAmobile, neither the company nor their competitors used any type of ML to perform the matching between users and advertisements. Therefore, during the execution of this project, it is expected to develop a novel learning method to perform this matching, fulfilling the Mobile Performance Marketing industry requirements, using OLAmobile to apply such method. Under this context, there is a set of scientific objectives and business- and technological-related requirements associated with this doctoral thesis.

1 https://promos.dsi.uminho.pt/en

The main goal of this thesis is to give a strong contribution to the body of knowledge regarding modern optimization of predictive models, using EC, with a particular focus in multi-objective tasks. Traditional ML approaches are not able to deal with these tasks, thus, using EC to optimize ML algorithms is the core of this thesis. To achieve this contribution, current state-of-the-art Modern Optimization algorithms applied to predictive models will be researched and evaluated under the Mobile Performance Marketing context, using the OLAmobile data. Our purpose is to develop a novel online learning model, capable of automatically learn form high-speed big data streams, taking advantage of multicore technology in order to be scalable and to guarantee a prediction response time inferior to 10 milliseconds. Furthermore, it is intended to use predictive models, automatically optimized by multi-objective EAs, providing accurate and real-time predictions, in order to perform the matching of users and product campaigns of their interest.

There are some important smaller objectives associated with this main objective and relative to Mobile Performance Marketing business and technological requirements, including:

- Extend the number and quality of used attributes for training the developed model. Currently, OLAmobile only uses a small set of fixed variables regarding user information (e.g., country, devices' operative system, mobile operator). During this research project, we intend to explore other attributes, namely advertisers and publishers information. Thus, by using a wider set of useful data, we expect to achieve better predictive performance.

- Design an efficient architecture for the parallelism approach. As already mentioned, the DSP must provide an advertisement to be presented within 10 milliseconds. Therefore, considering the computational limitations associated with the project computational server and the higher computational costs associated with EA, a smart use of hardware must be performed.

- Perform a robust and reliable evaluation of the developed model. To ensure the quality of our model, we intend to perform a robust benchmark comparing our algorithm with current state-of-the-art ML algorithms, considering at least two metrics: model predictive performance and computational cost.

- Provide explainable predictions to provide insights to OLAmobile experts. This objective requires the optimization of white-box predictive models (e.g., decision trees). By providing explainable predictions, the company decision makers are able to extract useful knowledge regarding their business.

- Enhance company's current metrics (e.g., CVR). This objective is directly related to model's performance, thus, performing a better matching between users and campaigns would reflect an improvement on its metrics and profits.

1.3 Research Methodology

Considering the nature of this research project, which involves the development of an artifact – a ML model – to address a specific problem, the adopted approach is Design Research. In particular, *Design Science Research Methodology for Information Systems (DSRM-IS)* was the chosen methodology for this project, and it consists in a set of techniques and analytic perspectives to develop *Information System (IS)* research. DSRM-IS, presented in Figure 2, is divided in 5 steps: Awareness of Problem, Suggestion, Development, Evaluation and Conclusion. Current section details how each of these steps were applied on this project's development.

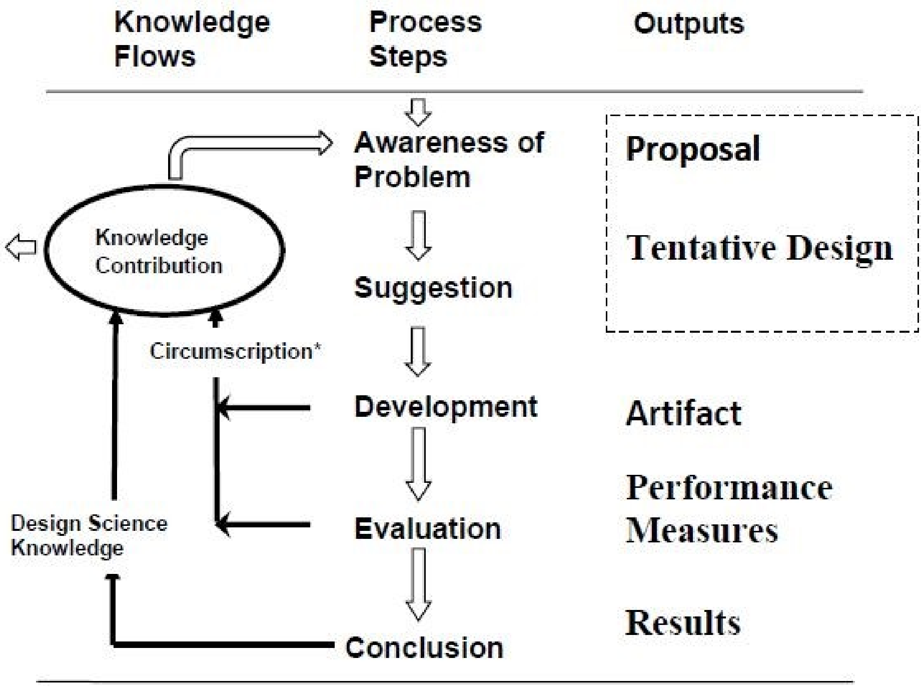


Figure 2: Design Science Research Process Model [taken from (Vaishnavi and Kuechler, 2004)].

The increased usage of mobile devices over the last years has led to vast opportunities for the Mobile Performance Marketing industry, which consists on displaying useful advertisements to users, on their mobile devices, aiming to lead them to perform a purchase. However, these opportunities also brought challenges regarding a real-time data treatment and mining. Currently, only nearly 1% of user clicks on a product campaign are converted into a sale, which is a tiny success percentage. Furthermore, companies operating on this marketing sector do not choose campaigns to be presented to users based on the possible interest, but, instead, on an eventual profit to the company. The methods used to perform this mapping between users and products are outdated, which justifies a small percentage of purchases face to the number of displayed campaigns. The first step of this project was the awareness of this issue, combined with the analysis of the current state of the art, which led to realizing that this situation could be improved if a different approach was used, and contributions could be made both in terms of business and body of knowledge. Hence, the formulated problem includes dealing with high-speed dynamic data at a real-time level by using non-traditional ML algorithms that can automatically give better advertisements suggestions to users, leading them to perform a purchase and, therefore, increasing some important companies metrics and, consequently, the profit. Moreover, exploring and implementing these algorithms could also lead to innovative contributions to the ML subject. Thus, a solution was thought and formulated, which leads to the next step of the methodology.

The second DSRM-IS step aims to suggest a solution for the identified problem, which should then be implemented and evaluated, aiming to increase the existing body of knowledge and contribute to solving the mentioned problem. Regarding this doctoral project, the suggested solution consists on using Modern Optimization algorithms to evolve predictive models, intending to improve their predictive performance on this type of marketing data, taking into consideration a set of limitations, including computational costs, real-time need for an accurate prediction, business needs and scientific contributions. Specifically, multi-objective *Genetic Algorithms (GAs)* were considered to create and train predictive models, such as DTs and *Neural Networks (NNs)*. This decision was taken based on state-of-the-art studies showing the flexibility and success of these approaches in different contexts (Mendes et al., 2002; Cortez and Donate, 2014).

After these two mentioned steps were accomplished, the next step concerns with the development of an innovative automatic ML modeling. It should be noted that DSRM-IS methodology is cyclic, which means that previous phases are not changeless or permanent, thus, can be (and were) modified when required. The development was produced in an iterative way, by testing different Modern Optimization techniques, *Data Mining (DM)* models and the combination of both, considering the defined requirements. In addition, different parallelism approaches were also considered, with the purpose of improving the system's efficiency (e.g., response time). For developing this system, the R and Python programming languages were used, due to their flexibility and success for different DM tasks. The output of this step was an IS artifact: the automatic ML algorithm, created using a multi-objective EA.

Throughout the development phase, the system was evaluated under different metrics (e.g., predictive performance, computational effort, prediction model complexity), such that it can be compared with relevant state-of-the-art baseline models. In order to do so, traditional ML algorithms were also implemented and tested. Then, a set of statistic tests were used to compare both implementations, aiming to increase the confidence on the results, producing a robust and fair comparison. In particular, we were able to produce an innovative and robust multi-objective result comparison method, detailed further on this document.

Finally, the last step of this methodology is achieved when the evaluated model accomplishes results that are considered satisfactory. On this project, satisfactory would mean that the developed system would have a better performance, statistically confirmed, when compared with the existing methods in a robust benchmark, fulfilling all established objectives. Since the DSRM-IS denotes a cyclic process, several iterations can be (and were) needed until we consider that success was achieved. Then, closing conclusions will be withdrawn, which often includes scientific publications. In effect, this project presented its research work in two JCR Q1 Journals and one conference, as detailed in the following section.

1.4 Contributions

This doctoral thesis was partially developed under the PROMOS R&D project, with a 3 year duration, in collaboration with OLAmobile, a company operating on Mobile Performance Marketing industry. In parallel with this thesis, another PhD thesis was developed, addressing the same issue and using the same data, but employing a different approach. Particularly, this thesis main goal relies on the contributions to the body of knowledge of Modern Optimization of predictive models, with a special focus in multi-objective optimization tasks, while the other one pays a particular attention to data preprocessing and the use of Deep Learning architectures for binary and ordinal classification tasks. In effect, both approaches were associated with OLAmobile, and therefore had to comply with the PROMOS project requirements. Nevertheless, during the project's first year, which also corresponded to this doctoral thesis first year, there was no available data regarding the specific OLAmobile use case. In effect, an *Application Programming Interface (API)* was being developed by the company with the purpose of allowing the collection of OLAmobile real-world data. Hence, we decided to start the PhD work with the exploration of a multi-objective EA for optimizing ML models, which is the main topic of this dissertation, giving particular focus to *Multilayer Perceptron (MLP)* NN and using nine time series datasets from several real-world domains. The purpose was to gain insights on the application of multi-objective GA for optimizing ML models for supervised learning tasks, as well as to develop a robust and reliable evaluation method for multi-objective ML tasks.

Under this context, this thesis first contribution is a set of neuroevolution methods for multi-step ahead time series *Prediction Intervals (PIs)* using *Non-dominated Sorting Genetic Algorithm II (NSGA-II)*. The purpose was to build PI, minimizing two different metrics regarding the achieved PIs: *Normalized Mean Prediction Interval Width (NMPIW)* and *Prediction Interval Coverage Error (PICE)*. Specifically, we proposed 6 *Evolutionary Artificial Neural Network (EANN)*-based *Multi-objective evolutionary algorithm Lower and Upper Bound Estimation (MLUBE)* methods, termed M2LUBET1, M2LUBET2, MLUBEXT, MLUBEXT2, M2LUBEXT and M2LUBEXT2. Furthermore, we also developed an innovative and robust evaluation method that uses a *Rolling Window (RW)* procedure with several iterations. Considering that GA are population-based and NSGA-II provides a set of Pareto-based solutions, analysing results from several iterations of RW is not an easy task. Therefore, our evaluation method aggregates similar solutions in one of the metrics and estimates Wilcoxon median values and 95% confidence intervals for each set of solutions. This way, the obtained graphics are more visually friendly and easily understandable, allowing a simple comparison between different *Multi-Objective Evolutionary Algorithm*

(MOEA) solutions. The work mentioned here is detailed in Chapter 3 and has resulted in the following publication of a conference paper and a journal article:

- Pedro José Pereira, Paulo Cortez, Rui Mendes. **Multi-objective Learning of Neural Network Time Series Prediction Intervals.** EPIA Conference on Artificial Intelligence 2017: 561-572.
- Paulo Cortez, Pedro José Pereira, Rui Mendes. **Multi-step Time Series Prediction Intervals Using Neuroevolution.** Neural Computing and Applications 32(13): 8939-8953 (2020).

After these contributions, several preliminary experiments were conducted by applying a *Grammatical Evolution (GE)* to design and evolve different supervised ML algorithms for a user CVR binary classification, using OLAmobile data. In the previous experiments, our base learner was a fixed-topology NN. Yet, considering that the other PhD thesis associated with the PROMOS R&D project was exploring Deep Learning architectures, we decided to try different base learners. Although it is not reflected on this document, several attempts were performed aiming to create and evolve Logistic Regression and Symbolic Regression models using GE. However, after several attempts that required several months of work, the obtained results were not considered satisfactory. Thus, we opted in a later PhD stage to change the target base learner, focusing on DTs. The motivations to choose this variety of model relied on its popularity in classification tasks and its easier interpretability when compared with other ML models (e.g., Random Forest). Moreover, we found some state-of-the-art works that used EAs to evolve DTs. Thus, our next contribution focuses on the application of GE-based DT for binary classification tasks (mobile user CVR prediction).

Chapter 4 meticulously describes our contributions for the Mobile Performance Marketing industry in the form of two novel approaches: *Multi-objective Grammatical Evolution Decision Tree (MGEDT)* and *Multi-objective Grammatical Evolution Decision Tree Lamarckian evolved (MGEDTL)*. The former uses a multi-objective pure GE method, while the latter takes advantage of Lamarckian Evolution process, using traditional DTs. Both these multi-objective approaches design and evolve *Evolutionary Decision Trees (EDTs)* that are able to deal with high-speed big data streams, taking advantage of parallelism techniques during the GE evaluation process, considering two relevant metrics: predictive performance and complexity. The first is measured in terms of *Area Under the Curve (AUC)*, which is flexible to unbalanced data, and the last is measured in terms of tree nodes, where lower values represent simpler solutions. Furthermore, we applied a random data sampling technique during this process, ensuring that our methods are not harmed by unbalanced datasets, as the ones provided by OLAmobile. By using our previously developed evaluation method, we performed a robust benchmark comparing our approaches with different ML algorithms, namely traditional DT, Random Forest and Deep Learning, achieving quite interesting results. Although we do not directly provide a campaign to a user, our EDT, similarly to regression trees, outputs a numerical probability of a user to purchase a given product.

With this approach, we can test from around 5 thousand (with MGEDTL) to 2 million (with MGEDT) different campaigns within the OLAmobile 10 milliseconds time limitation. Then, the campaign with the highest probability to originate a conversion can be selected and displayed to the user. Moreover, the work developed in Chapter 4 originated a Python module named **evoltree**, available on PyPi[2] and GitHub[3], whose user manual can be found in Appendix A, and also led to the following journal article publication:

- Pedro José Pereira, Paulo Cortez, Rui Mendes. **Multi-objective Grammatical Evolution of Decision Trees for Mobile Marketing User Conversion Prediction.** Expert Systems with Applications 168: 114287 (2021).

1.5 Thesis Organization

This dissertation is composed by 4 main parts, which include 5 chapters and 1 appendix. The first part is composed by Chapters 1 (Introduction) and 2 (Background), providing, respectively, an introductory description of the addressed problems during this thesis and the relevant state-of-the-art. In particular, Chapter 2 is divided in two main contents. The first concerns with a theoretical introduction with a few relevant concepts regarding the subjects addressed during this project, namely Big Data (Section 2.2), Data Mining (Section 2.3) and Modern Optimization (Section 2.4). During these sections, the main concern is to provide an introduction to the mentioned topics. The second Chapter 2 content consists of a state-of-the-art analysis of recent and important practical studies related with the application of learning methods to different domains, as well as describing the Mobile Performance Marketing industry. Specifically, Section 2.5 describes how optimization algorithms are used to create and/or evolve different predictive algorithms, with a particular focus in the Neuroevolution and *Genetic Programming (GP)* and GE subjects; Section 2.6 describes the industry studied during this PhD, its main stakeholders and commonly used metrics by companies as OLAmobile; Section 2.7 presents current studies on the Mobile Performance Marketing and similar areas, using ML algorithms and, in some cases, Modern Optimization, aiming to identify research gaps; finally, Section 2.8 presents the final conclusions, as well as the research path we intend to follow.

In terms of the second part, composed by Chapters 3 and 4, it describes the main experimental work developed during this doctoral project. Chapter 3 is based on a journal article (Cortez et al., 2020), published during this PhD project. During this chapter, we detail our proposed multi-objective Neuroevolution methods for multi-step ahead time series PI. This work is a continuation from a previous work (Pereira et al., 2017) and it was developed in an initial phase of this project, during which there was not any available data from OLAmobile. Thus, it was developed with the purpose of gaining insights

2 https://pypi.org/project/evoltree/
3 https://github.com/p-pereira/evoltree

with the multi-objective modern optimization of predictive models. Moreover, during this chapter we proposed an innovative evaluation method for multi-objective results and that was used later in the work developed in the next chapter. Turning to Chapter 4, it is based on our core journal publication regarding the contributions in the Mobile Performance Marketing industry (Pereira et al., 2021), including work also developed during this PhD. More specifically, we developed two novel multi-objective approaches for designing and evolving EDT, using predictive performance and complexity as goals. Both approaches use a multi-objective NSGA-II-based GE python implementation, one with pure GE method and one including a Lamarckian Evolution. The EDT were evaluated using the innovative method developed in Chapter 3 work, assuming real-world Mobile Performance Marketing datasets provided by OLAmobile. Furthermore, this chapter provides a robust benchmark, comparing results achieved by our proposed approaches with 3 ML algorithms, namely traditional Decision Trees, Random Forest and Deep Learning.

The third part comprehends Chapter 5 (Conclusions). Here, the main conclusions of this doctoral project are discussed, as well as its limitations and possible directions of future work. Finally, the last part (Appendix A) includes a user manual for the developed Python module, regarding the work presented in Chapter 4.

2

BACKGROUND

2.1 Introduction

The present chapter reviews the relevant state-of-the-art works, concerning the use of Modern Optimization for designing predictive models, using Big high-speed Data, applied to the Mobile Performance Marketing industry. Thus, it starts explaining the search strategy, followed by a theoretical introduction where the main concepts and models are addressed, then explaining the area under study and, lastly, analyzing developed work on this area, aiming to find a research gap that we intend to fulfill. Several studies were proposed regarding Data Mining applications for numerous subjects, including marketing, economics, finance, etc., which simplified the process of finding relevant literature about it. Hence, this literature review was divided in two main stages: theoretical introduction to the subject, where main concepts are described, and practical studies, which describes the relevant works conducted.

For the theoretical introduction, the majority of used references were mainly in books. Part of the used books were provided by the supervisors (e.g., (Cortez, 2021)), and the remaining were found online in databases, such as Google Scholar[1] and Google Books[2]. The used terms for the bibliographic search were: "Big Data", "Data Mining", "Knowledge Discovery from Databases", "Data Mining Models", "Neural Networks", "Modern Optimization", "Metaheuristics", "Neuroevolution", among others. Found references on this search were used to write the following sections and related with: Big Data (Section 2.2), Data Mining (Section 2.3) and Modern Optimization (Section 2.4). Although it is not directly reflected in the mentioned sections, some Massive Online Open Courses (MOOCs) were attended as a supplement for this first stage, with the purpose of providing important insights related to a few relevant subjects, including Neural Networks, Metaheuristics and Multi-Objective Optimization. The used platforms to attend these courses were Data Camp[3] and Coursera[4].

Regarding the second stage of this literature review search, the purpose was to find practical studies concerning the optimization of predictive models for big data sets, particularly, in the marketing industry. Furthermore, this search was extended to similar areas due to the lack of proposed stud-

1 https://scholar.google.pt
2 http://books.google.pt
3 http://datacamp.com
4 http://coursera.org

ies in this area. The used databases for this search were Google Scholar, Science Direct[5], Scopus[6] and IEEE Xplore[7]. Within these databases, the used terms were more specific, including: "Evolving Neural Networks", "Neuroevolution", "Genetic Algorithms and Neural Networks", "Evolutionary Computation and Neural Networks", "Big Data and Neural Networks", "Data Streams Prediction", "Mobile Performance Marketing", "User Intention Prediction", "NEAT", "NEAT Supervised Learning", "Genetic Programming", "Grammatical Evolution", "Island Model", among several others. In each search, when possible, filters were applied regarding the study year, in order to select the more recent works (mostly in the last 10 years). Moreover, it was also applied a filter to select journal papers and Web of Science Journal Rank[8] was used to select the papers from the best journals. Finally, based on the abstract and conclusions sections, and sometimes the results, the retrieved papers were selected or discarded. This search allowed to find related studies in terms of optimization of machine learning algorithms, used in Sections 2.5 and 2.7, as well as regarding mobile performance marketing and similar industries, used in Section 2.6. Based on the analysed studies, final conclusions are withdrawn and a research gap is identified in Section 2.8.

2.2 Big Data

Several advances in *Information Technologies (IT)* have allowed the gathering and storing of virtually any type of data with a minimum effort, which leads to a huge increase in terms of data volume to be stored. There were several attempts to estimate and predict data volume and its growth, given its importance on multiple fields.

In 2016, Dhaenens and Jourdan estimated that, by 2020, 40 zettabytes (40 trillion gigabytes) of data would be produced. However, recent forecasts (Holst, 2021) show that this amount was surpassed in 2019. Furthermore, Holst estimates that this value will increase more than 4 times by 2025. Social networks and traffic management, in a smart city context, are examples of systems that generate vast volumes of data in a matter of seconds. For instance, according to Domo's Data Never Sleeps 8.0 report (Domo, 2021), in 2020, every minute 500 hours of videos were uploaded to Youtube, 41.7 million messages were shared on WhatsApp, 147 thousand photos were uploaded to Facebook and 3.8 thousand dollars were spent on mobile apps.

The produced data is not anymore limited to text and numerical values recorded in structured databases, but also huge amounts of unstructured data, in several formats (e.g., as documents, images, audio files and even videos), generated by a countless number of devices, including smartphones, sensors, GPS devices or cameras. This huge amount of data, generated by several devices in a great velocity, leads to a subject well known as Big Data.

5 http://sciencedirect.com
6 http://scopus.com
7 http://ieeexplore.ieee.org
8 https://mjl.clarivate.com/home

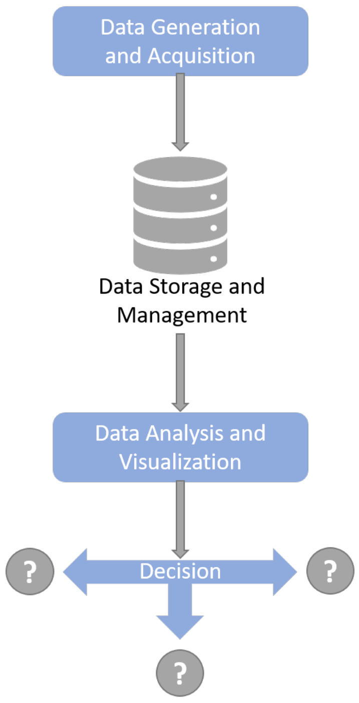

Data Generation and Acquisition
Data Storage and Management
Data Analysis and Visualization
Decision
?
?
?

Figure 3: Big Data Process [adapted from (Dhaenens and Jourdan, 2016)].

There are many definitions for Big Data, however, the most popular and frequently used (Loshin, 2013) belongs to Gartner and states: "Big Data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making" (Gartner Research, 2012). The definitions change with authors, yet, there are 3 dimensions (3 V's), that seem to be agreed by the majority, as Big Data common properties: Volume, Velocity and Variety. Volume is related to the amount of data that is created and collected, which usually shows an exponential growth in terms of size; velocity concerns with the dynamics associated with the data volume, i.e., data streams coming in different speeds and different quantities, often needing to be processed and treated in nearly real-time; variety involves having several sources generating different types of semi-structured or unstructured data, such as text, log files, audio, among others (Morabito, 2015; Dhaenens and Jourdan, 2016). More recently, two more V's (dimensions) have been considered: Veracity and Value. The former addresses the data quality and the latter the additional value produced from collected data, which is the main interest of collecting it. Thus, Big Data does not refer only to a vast volume of data, but mainly with its complexity (Dhaenens and Jourdan, 2016). If data is coming at a regular velocity and in a structured way, complexity would not be considered higher when compared to unstructured data arriving in an unpredictable speed and with lower volume.

According to Dhaenens and Jourdan (2016), there are 3 main challenges regarding Big Data (Figure 3): data generation and acquisition, data storage and management and data analysis. The first challenge does not address how to generate and acquire data, since recent technology (e.g. digital sensors) allows doing it easily. Instead, it concerns with what information is interesting to record or measure. Data storage and management becomes a challenge when relational databases are no longer efficient, mostly due to the lack of capability to adapt to Big Data, i.e., this type of databases cannot deal with Big Data specificities, such as scalability (physical infrastructure limitations), variety of data (including unstructured data) and data velocity (non-synchronous acquisition). Consequently, non-relational database technologies have been developed, such as NoSQL, that do not depend on static tables. Examples of these technologies are key-value pared databases, document databases – structured using *JavaScript Object Notation (JSON)* –, columnar or column-oriented databases (such as HBase), among others. Finally, the last challenge is concerned with knowledge extraction from data, which can be divided into 3 levels: reports, multi-level analysis and complex analysis.

This research project does not address problems concerning data generation and acquisition, since the used data is provided by the Mobile Performance Marketing organization, which will be described in Section 2.6. Considering this Big Data context, data storage and management will inevitably be a technological problem to take into account. Yet, it is not considered our main focus. Instead, the main objective is focusing the third Big Data challenge: complex analysis. This subject aims to discover unknown patterns in data, which is often known as Data Mining, and will be explored throughout this document.

2.3 Data Mining

Considering the vast amount of data existing nowadays, it is important to give it a meaning, i.e., to extract knowledge and useful information from it, which is the purpose of *Data Mining (DM)*. As a scientific interdisciplinary subject, DM does not have a globally agreed definition. Nevertheless, it is commonly defined as the process of extracting knowledge from a set of data (Hand et al., 2001; Han et al., 2011; Leskovec et al., 2014; North, 2012; Dhaenens and Jourdan, 2016; Sugiyama, 2016), where knowledge if often represented in terms of models or patterns. In some cases, DM is considered a synonym for *Knowledge Discovery from Databases (KDD)* and for *Machine Learning (ML)*, in other cases ML is considered a stage of DM process, which is considered an essential mathematical step of KDD. In this project, it will be considered that DM and KDD are both synonyms for the process of extracting interesting, useful and valid patterns from datasets supporting decision-making, with ML being a key technology used to achieve it (North, 2012; Sugiyama, 2016).

The term DM, which can be traced to the late 1980s (North, 2012), is an analogy to gold extraction, with knowledge being the gold and data being the rocks (Han et al., 2011) and it consists in 6 main steps: data cleaning, data integration, data selection, data transformation, data mining/modeling, model evaluation and knowledge presentation. In 1999, a standardized approach to Data Mining known as *CRoss-Industry Standard Process for Data Mining (CRISP-DM)* (North, 2012) emerged, which is often used for DM projects and that also consists in 6 steps: business understanding, data understanding, data preparation, modeling, evaluation and deployment.

ML can be defined as a set of methods for data analysis and modeling that are based on *Artificial Intelligence (AI)* and the idea behind it is to try to imitate natural intelligence, where learning tends to be based on the use of examples, counter-examples and exceptions (Nettleton, 2014). Depending on learning objectives and the type of data, there are three main types of ML that can be used: Supervised Learning, Unsupervised Learning and Reinforcement Learning.

2.3.1 *Supervised Learning*

Supervised learning is the most common type of ML and the way it works is similar to a teacher-student relation, where the computer is the student and the user is the teacher (Sugiyama, 2016). Briefly, the user feeds the computer with examples and counter-examples so that the computer learns a mapping between input and output values with the purpose of obtaining generalization ability, so it can map unseen inputs to its expected outputs. Considering a face recognition example, giving some examples of pictures with faces and counter-examples without faces (training set), the computer should learn to recognize a face. When new pictures are given to the computer (test set), it should be able to generalize and know whether there is a face in the picture or not.

Within supervised learning, there are two main tasks, classification and regression, depending on the target. The first is used when the target is a class (discrete value) and the later when the target is a numerical value. A classification example can be to classify wine quality (good or bad) based on its components, where the target is “good” or “bad”. Considering the same case, a regression example would be to predict the wine quality in a numerical value. Choosing a ML model often depends whether the target is a class or numerical. However, there are models that, although they are more commonly used in one task, can be used in both, namely *Decision Tree (DT)*, *Support Vector Machine (SVM)* and *Neural Network (NN)*. A brief description of these models is provided further on this document.

2.3.2 *Unsupervised Learning*

Unlike supervised learning, unsupervised does not have labeled data, i.e., it is not possible to directly map an input to its output. Therefore, the idea of a teacher-student relation is not applied here. On the other hand, the purpose of this kind of learning is to find different patterns of data, including the aggregation of instances in different groups (or clusters) based on their similarities (Torgo, 2010). This process is called clustering, but there are other unsupervised learning tasks, such as outlier and anomaly detection.

Unsupervised learning can also be useful when complementing supervised learning, for instance by using as a preliminary approach to analyze the data. Moreover, organizations can use clustering for grouping their clients into clusters and understanding similarities and patterns that are not easily detected. K-means and self-organizing maps (a special type of neural networks) are algorithms that can be used for these tasks.

2.3.3 *Reinforcement Learning*

Reinforcement learning describes a learning problem characteristic of autonomous agents interacting in an environment where the algorithms seek to learn how to map from states to actions by maximizing the reward received over time (Sammut and Webb, 2017). Returning to the teacher-student idea mentioned in supervised learning and applying it to reinforcement learning, the teacher evaluates the student's behavior and gives feedback about it (Sugiyama, 2016), instead of giving him/her the correct answer (class or numerical value that should had predicted). A common example used to describe this kind of learning is driving a car, where the student has an action (turn the wheel to left, for instance) and his action will be translated in a change in the environment (the car moving), which will either be rewarded or penalized depending on whether it was the correct action or not.

This sort of approach is often used for automatic game players, where the player receives a penalty or reward when performing a given move and, therefore, ends up learning how to play, even without previous knowledge about the game. This type of learning can also be applied to decision support by penalizing or rewarding the algorithm for each decision it makes. NN are often used in reinforcement learning by combining them with evolutionary algorithms, for example. These approaches will be described in Section 2.5.1.

2.3.4 *Offline and Online Learning*

So far, the mentioned learning approaches were described in a static perspective, which means that the algorithms learned from data that was already stored. This approached is named offline learning, also known as batch learning, and it does not have to do with the learning approach (e.g., supervised), but instead with the used data and its environment, which does not require a continuous learning process. In offline learning, the algorithm accepts a single dataset, corresponding to a set or sequence of observations, it produces its model and does no further learning (Sammut and Webb, 2017). However, in a Big Data complex environment, this option is often not reliable, since the data is continuously arriving with new variables, new values and different contexts, and offline learning algorithms do not quickly adapt to this constantly changing environment. Due to these limitations, a different approach is needed, and it is called online learning.

In online learning, the data is continuously arriving in data streams, this means that training samples are provided in smaller sets (e.g., one observation at time) in a sequential manner, and the algorithm needs to make predictions or choices on it. After a set of predictions or choices, models are retrained, aiming to learn new changes on context or environment from newer observations. This kind of approach is useful when the training sample is too large and it cannot be stored (Sugiyama, 2016) or when dealing with complex and constantly changing environments, such as the Mobile Performance Marketing domain.

2.3.5 *Learning Algorithms*

Depending on the task, there are several possible choices concerning the used algorithms. In effect, there are several algorithms that can be used in supervised learning (both for regression and classification tasks), such as DT, SVM and NN. Furthermore, there are several NN adaptions, that can be used for supervised, unsupervised and reinforcement learning. A brief description of these models is given in this section.

Decision Trees

Although it can also be used for regression tasks, DT, also called classification trees, are very popular for classification tasks, since they tend to provide an human understandable explanation for their patterns, i.e., it is relatively simple to understand why a given set of inputs were mapped to an output (Dhaenens and Jourdan, 2016). As the name suggests, a DT is composed by decision branches, which are read from top to bottom, and contain a true/false condition. The decision to follow a given path is based on that condition's result and, in case it is a classification problem, the path should lead to a class. Figure 4 presents a very simple example of a decision tree, with the top rectangle being the root node, the other rectangle being an internal node, the arrows being branches and the circles being leaf (or terminal) nodes.

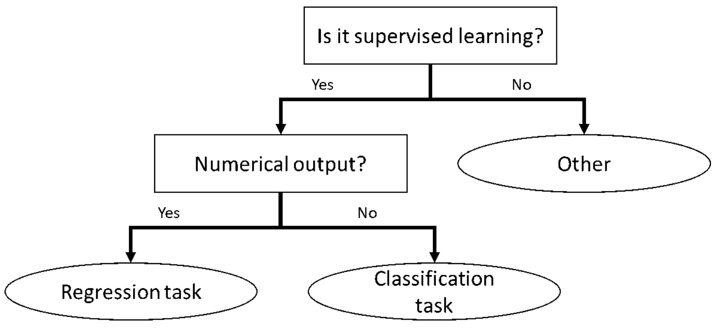


Figure 4: Example of a Decision Tree.

In a few words, constructing a decision tree consists in recursively dividing a training set until each division contains entirely, or primarily, examples from one class and each internal node has a split point, which determines how the data should be divided further (Turban et al., 2010).

Support Vector Machines

As a result of its strong theoretical background and successful application to several domains, for both regression and classification tasks, SVM have the been center of attention for different research communities (Torgo, 2010). This method converts the original data into a high-dimensional space, so it can be possible to apply linear models to obtain a separating hyperplane. The mapping between the original and the new high-dimensional space is done using kernel functions.

Considering Figure 5 and a binary classification example, the left image contains two lines: A and B. Line B is closer to critical boundary instances (support vectors) from both classes (white and red points), so, when new examples arrive, it is more likely to fail classifying them when compared with line A, this means, the new points are more likely to fall on the wrong side of the separator (Battiti and Brunato, 2017). The idea behind SVM is to insert the hyperplane (right image, blue rectangle) between the support vectors of each class, separating them, and then maximize the margin, which corresponds to the distance between line A to the support vectors.

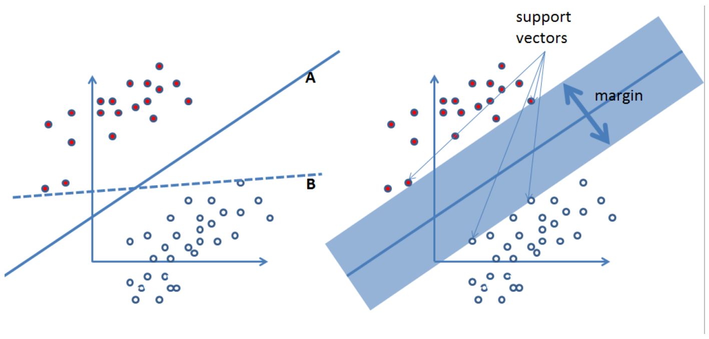


Figure 5: Basis of Support Vector Machines [taken from (Battiti and Brunato, 2017)].

Although SVMs training phase can be extremely slow, it is highly accurate, it can model complex nonlinear decision boundaries and it is less prone to overfitting (Han et al., 2011). Besides, a SVM key point is that is maximizes the geometric margin and minimizes classification error simultaneously (Dhaenens and Jourdan, 2016).

Neural Networks

NN, or *Artificial Neural Network (ANN)* are a mathematical model that seeks to replicate the human brain's processing system and are capable of learning with experience. The human neural system is composed by 100 billion computing units (called neurons) and about one quadrillion connections (Battiti and Brunato, 2017). Traditional ANN are composed by three or more layers, each containing a group of neurons (or nodes) interconnected among them by using connection weights (a numerical value) that is changed during the NN learning phase. The human brain intelligence is due to its massively parallel neurons, whose system is commonly known in the jargon by network architecture or topology, and a proper design of an ANN can offer significant improvements in its learning (Ojha et al., 2017).

A NN topology has to do with the number of layers, the neurons present in each layer and its connections. For supervised learning tasks, NN have input and output layers and the number of nodes in each is dependent on the number of variables (also known as features) – where the training data is fed to the network – and the number of outputs (targets), respectively (Torgo, 2010). The remaining layers are called hidden layers and it is the user who defines the number of hidden layers, such as the number of neurons in each layer. This number is very context-dependent, i.e., there is not a correct number of layers and nodes to be used in all problems, it depends on the data and problem complexity. Thus, based on the number of layers, nodes and the way they are connected, NN have different designations.

The simplest form of a NN model is a *Single Layer Perceptron (SLP)* – proposed in the 50s – which is only composed by an input and an output layer. From a ML perspective, the model is only capable of making linear mappings from a set of inputs to a set of outputs (Neukart, 2017; Ojha et al., 2017). A *Multilayer Perceptron (MLP)* NN was proposed in the 70s to deal with this limitation by including one or more hidden layers between the input and output ones. These two types of ANN can be categorized as *Feedforward Neural Network (FNN)*, which means that connections between neurons does not form a cycle and go from input to output layer, in this "forward" order. *Radial Basis Function (RBF)* NN (proposed in the 80s) and SVM (proposed in the 90s) are special types of three-layer FNN, capable of solving both regression and classification tasks, for supervised learning (Ojha et al., 2017). Furthermore, deep neural networks (or deep learning) are also a successful kind of NN topology, recently gaining a huge popularity due to its success in diverse ML competitions, that can have more than thousands of neurons and layers. However, this can bring great impact in the learning speed. There are also FNN that can be used in unsupervised learning tasks, such as Kohenen's *Self-Organized Map (SOM)* and learning-vector-quantization, both proposed in the 80s and both with only two layers, capable of solving pattern recognition and data compression tasks (Ojha et al., 2017).

Another NN architecture is called *Recurrent Neural Network (RNN)*, or feedback network model and, unlike FNN, these networks have feedback connections, this means, connections between nodes can form cycles (Ojha et al., 2017). Hopfield Network and Boltzmann Machine (both proposed in the 80s) are two examples of RNN, both being good at the application of memory storage and remembering input-output relations (Ojha et al., 2017). Figure 6 presents some examples of the distinct NN types, with grey circles representing input nodes, yellow circles hidden nodes, orange circles output nodes and arrows and straights representing connections between them.

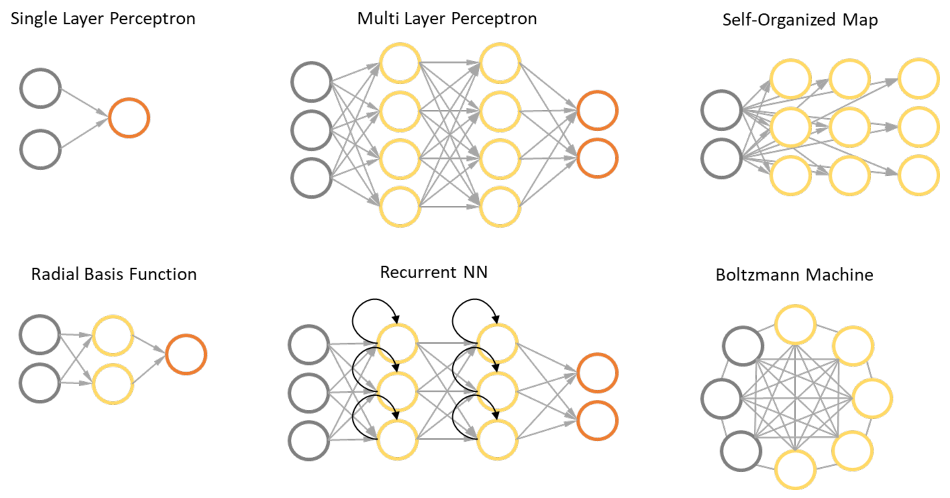


Figure 6: Examples of Neural Network topologies [adapted from (Santos, 2016)].

Considering a FNN neuron, the weighted sum of inputs, and optionally a bias value is added in order to shift this sum, yields a value known as neuron's stimulus or inner potential (Volna, 2010). Thus, a traditional FNN's learning phase corresponds to the optimization of connection weights (and optionally bias values) to minimize a predefined error metric. This optimization is often performed using an algorithm called Backpropagation. Typically, when this learning phase occurs the topology is already fixed and weights and biases are the only values updated. Yet, there are recent studies that consider both weight and topology as an optimization problem, which can bring great advantages, as can be seen in Section 2.5.

2.4 Modern Optimization

Before explaining the concept of modern optimization, it is important to understand its origins. Optimization is a fundamental topic of operations research, which consists on applying scientific methods and techniques to decision making problems and in establishing the best or optimal solutions. Operations research beginnings can be traced to the 40s, during the second world war, where the British military needed to allocate their scarce and limited resources (airplanes, radars, submarines) to several activities and they called upon a team of mathematicians to develop methods for solving this problem in a scientific way. The result of this research were a set of methods, nowadays known as classical techniques, such as linear programming, that helped Britain win the air battle (Rao, 2019).

Optimization is the process of obtaining the best result under a set of circumstances and it consists on finding the conditions that give the maximum (or minimum) value of a function (Rao, 2019; Cortez, 2021). Currently, several real-world tasks can be viewed as optimization problems and, thus, optimization techniques are used in several domains, namely Engineering, Finance, Marketing and Science.

In the last decade, a new class of optimization methods that is conceptually different from classical techniques has emerged and is called Modern Optimization (Michalewicz et al., 2006; Cortez, 2021), also known as Metaheuristics or Modern Heuristics.

This type of optimization algorithms, contrasting with classic ones, are general purpose solvers, which means that they can be applied to a wide range of problems (often complex) (Cortez, 2021).

In particular, modern optimization techniques often do not guarantee that the optimal solution will be found, but instead they achieve high-quality solutions that satisfy the user's needs with less computational effort. There are only two things to be specified when using these methods: representation of the solution and evaluation function (Cortez, 2021). The former concerns with the search space and its size, and there are several ways of representing solutions, namely binary, integer and real values, characters, matrix, lists and even a combination of those; the later consists in defining a way of measuring and comparing the quality of solutions, which is often translated into a numerical value (the objective) that is intended to be maximized (or minimized) by the algorithm. After solving these issues, the algorithm will start its search for a solution (or a set of solutions) and the way to do this depends on the type of search, that can be blind, local or population-based.

2.4.1 *Blind Search*

The type of search defines how the method will generate the next solution in order to improve its objective. When assuming a full blind search approach, it will exhaustively search for all possible alternatives, without using knowledge from previous solutions (Cortez, 2021). This kind of approach is not feasible for most real-world problems since the search space is continuous or too large. Blind search can only be applied to discrete search spaces and can be easily encoded in two ways (Cortez, 2021). The first is by defining the full search space in a matrix, where each solution corresponds to a row and the algorithm sequentially tests each row. The second is by recursively organizing the search space as a tree, where branches denote possible values for variables and all solutions appear at the leafs.

Considering the search space as a tree, the algorithm can be depth-first, which starts at the tree's root and goes through each brand as far as possible before backtracking, or breadth-first, which also starts on the root but searches all succeeding nodes at the first level, and then the next succeeding nodes, until the nodes from the last level, or a stop condition, is found (Cortez, 2021). As pure blind search is not feasible in most of the optimization problems, the algorithms are usually adapted with the purpose of reducing the search space by, for instance, limiting the number of levels that the algorithm can search in a tree. Grid Search and Monte Carlo Search (also known as random search) are popular examples of these methods with a reduced search space.

2.4.2 *Local Search*

Global optimum is the best solution in the entire search space and it can be found if the size of the search space is small or if the whole space is tested. However, this approach is often considered unfeasible, so the concept of neighborhood is used. It consists on searching for the next solution in a subset of the search space close to the previous solution. Yet, this approach can cause the algorithm to be trapped in a local optimum, which is the best solution in the neighborhood, instead of finding the global optimum (Michalewicz et al., 2006). Local search, also called single-state search, consists on searching the neighborhood of a solution, instead of searching the entire search space.

Local optimization techniques bring, however, a trade-off between the size of the neighborhood and the feasibility of the search (Michalewicz et al., 2006). In one hand, if the neighborhood's size is small, the algorithm may be able to search all solutions quickly, thus, increasing the probability of being trapped in a local optimum. On the other hand, if the neighborhood is too large, although it reduces that probability, it could become impossible to compute given the amount of solutions to be tested.

Unlike blind search techniques, modern optimization ones (either local or population-based search) use the previous solution for generating the next one, which is called guided search. The first solution can be generated using previous knowledge about the optimization problem or randomly (Cortez, 2021). What differentiates the algorithms is how the next solution is generated and what is kept from the previous one. Examples of local optimization algorithms are Hill Climbing, Simulated Annealing and Tabu Search. Hill Climbing stands for climbing up a hill, assuming that is solving a maximization problem, until a local optimum is found. Simulated Annealing, proposed in the 80s, is inspired in a metallurgy process that consists in heating a material and making its temperature to decrease slowly to allow atoms having a good organization that would correspond to an optimum (Dhaenens and Jourdan, 2016). Tabu Search, also proposed in the 80s, is similar to Hill Climbing, yet, in some cases, it accepts non-improving neighborhoods in order to escape local optimum.

2.4.3 *Population-Based Search*

So far, the presented methods used only a single search point in each search. Population-based search, on the other hand, uses a pool of competing solutions called population, which tends to require more computational effort when compared with a simple local method, but also tends to reach more global optimization values (Cortez, 2021). Population-based methods tend to explore more distinct regions of the search space and most of them are based on analogies to natural concepts (similar to Simulated Annealing). This type of methods can be divided in two main categories (Dhaenens and Jourdan, 2016): *Evolutionary Algorithm (EA)* and Swarm Intelligence.

Evolutionary Algorithms

Evolutionary algorithms belong to the field of *Evolutionary Computation (EC)*, which addresses the computational methods inspired in the process and mechanism of biological evolution by means of the natural selection theory by Charles Darwin. Natural selection accounts for the variety of life and its suitability (adaptive fit) for the environment and its evolution mechanisms describe how evolution occurs through the modification and propagation of genes (Brownlee, 2011). These methods share a similar structure and have a biological terminology associated with it, some of them stated as follows, according to Luke (2013):

- **Individual** - a candidate solution;
- **Child and parent** - child is the adjusted copy of a candidate solution (its parent);
- **Population** - pool of individuals;
- **Fitness** - evaluation function;
- **Evaluation or fitness assessment** - computing the fitness of an individual;
- **Selection** - process of selecting individuals according to their fitness;
- **Mutation** - performing slight changes to individuals;
- **Crossover** - generation of children through the combination of two or more parents;
- **Breeding** - producing one or more children from a population of parents;
- **Genotype, genome or chromosome** - individual's data structure;
- **Gene** - slot position of a genome/chromosome/genotype;
- **Allele** - value for a gene;
- **Phenotype** - how individual operates during evaluation;
- **Generation** - one cycle of evolution (selection, breeding, crossover, mutation) and evaluation.

There are several algorithms within the EA class and they differ from one another in the type of solutions and the way they are generated. The most popular algorithms from this class are *Genetic Algorithm (GA)*, *Differential Evolution (DE)*, *Estimation of Distribution Algorithm (EDA)*, *Genetic Programming (GP)* and *Grammatical Evolution (GE)*. GAs, the most used within the EAs, were proposed in the 70s and initially worked only with binary representations (Dhaenens and Jourdan, 2016). These algorithms are inspired by population genetics and evolution at population level, where individuals contribute with their genetic material, proportional to their suitability to the environment, in form of offspring (Brownlee, 2011). GA's objective is to maximize the fitness of the population, by selecting a set of successful individuals, based on their fitness, for creating descendants that will be inserted into the new population. Currently, GA is no longer limited to binary representations, so it can deal with other type of representations (e.g., numerical values) and it can be used in DM problems, as it will be shown further on this document.

DE, developed in the 90s, has been successfully applied to solve continuous optimization problems (Cortez, 2021). Similar to GA, this method evolves a population of individuals, where each individual is encoded by a vector of real values. The main difference when compared to GA are the mutation and crossover operators. Mutation in DE consists in the use of arithmetic operators to generate new solutions. The advantage of using DE is that only few control parameters are required, so it is easy to tune (Dhaenens and Jourdan, 2016).

EDAs, also known as *Probabilistic Model-Building Genetic Algorithm (PMBGA)*, were proposed in the mid-90s and are optimization methods that combine ideas from EC, ML and Statistics (Mühlenbein and Paass, 1996). EDA iteratively makes a probabilistic estimation of promising solutions (good individuals), using them to create new individuals, instead of using crossover and mutation operators as GA and DE, and it can lead the search towards interesting areas from the search space. These algorithms can perform an effective search and have been applied to a wide range of problems, however, several questions are still open regarding their behavior.

GP, proposed in the 90s, is a modern optimization method that, instead of numerical optimization, is used for automatic programming or discovering mathematical functions (Koza, 1993). It uses a tree representation, with the leaves representing variables and constants and the internal nodes representing functions or operators. In GP, individuals are themselves programs. As GA, GP is often used for DM and ML tasks, however, it has a high computational cost due to its high search space.

Finally, GE is also a global optimization technique within the class of EA, quite similar to GP since it is also used for evolving computer programs (Ryan et al., 1998). Yet, unlike GP that performs the evolutionary process on the actual programs, GE does it on variable-length integer chromosomes, similar to classic EA. In GE, a mapping process is employed to generate programs in any language by using integer chromosomes to select production rules in a defined grammar. The result is the construction of a syntactically correct program to be evaluated by a fitness function (O'Neill and Ryan, 2003). Although GE solutions are syntactically correct, they are often more complex than needed. This phenomenon, known as bloat, leads to an unnecessary increase of the computational cost.

EA is not limited to the mentioned examples. Although these are the most widely used algorithms, there are other interesting ones within this class that work with parallelism. In particular, distributed GA is an example of this type of algorithms, that are designed to divide a population across computer networks, or even computational units, allowing to process different individuals in parallel and trading information between each processing unit (Brownlee, 2011). These type of algorithms are often termed Island Models, considering each processing unit as an island.

Swarm Intelligence

Besides EA, there is another important class of naturally inspired algorithms termed Swarm Intelligence, which refers to algorithms or distributed problem-solving devices inspired on the collective behavior of insect colonies and other animal societies, such as ants, bees, bats, fish, birds, among others. Swarm Intelligence algorithms work with a set of simple particles (individuals) that interact with each other and with the environment, leading to a global and self-organized behavior. Although several different analogies were proposed, such as bee colony optimization, artificial immune systems and bat algorithm, the two most famous ones are Ant Colony Optimization and Particle Swarm Optimization methods (Dhaenens and Jourdan, 2016).

Ant Colony System belongs to *Ant Colony Optimization (ACO)* methods, proposed in the 90s, and it is based on the social behavior of ants, specifically, on their communication for finding the path to food. Briefly, real ants walk randomly in different directions in the environment in order to find food and, once one of them finds it, starts releasing pheromone particles on the path from the food source to the nest. Eventually, other ants find the pheromone and start going also through that path, laying down more pheromone, ending up with the whole population of ants following that path (Brownlee, 2011). Artificial ants have a similar behavior, where each ant (individual) is evaluated and is associated with a type of pheromone-based fitness, where higher levels of pheromone implies better performance. Finally, ants will deposit pheromone on the path they travelled, based on the quality of the solution found. New solutions will be created influenced by the pheromone in those paths (Michalewicz et al., 2006). Unlike the GA, these methods do not use mutation and crossover operators to generate new solutions, but instead the levels of pheromone on the edges of the graph.

Particle Swarm Optimization (PSO), also proposed in the 90s (Kennedy and Eberhart, 1995), is inspired by the social behavior of some animals (e.g., fish, birds) of widely searching for food or provisions. In a metaphoric perspective, particles in the swarm (individuals in the population) follow the swarm's fitter members biasing their movement toward historically good areas on the environment, the search space (Mendes et al., 2004). Regarding the PSO algorithms, it consists on randomly generating autonomous particles in the search space, associating a velocity to each, its location in the search space, memory to keep track of its best solution and the global best. Then, the optimization process changes the velocity of each particle toward the individual and global best position, considering the social influence between particles. PSO has also been used for DM tasks (Mendes et al., 2002), as it will be shown further on this document.

2.4.4 *Multi-objective Optimization*

So far, despite their differences, all the addressed optimization algorithms aim to optimize a single objective. However, more complex situations often demand a different approach, which includes more than one objective to be optimized. For instance, minimizing the number employees in a store and maximizing the store's profit, or minimizing both a distance to travel and the time spent. This approach is termed multi-objective, which consists in simultaneously minimizing (or maximizig) two or more different objectives. Often, this approach includes dealing with a trade-off between them, i.e., considering a minimization task for two objectives, minimizing one of them will increase the other's value, and vice-versa. There are three approaches to deal with multi-objective optimization problems (Cortez, 2021): weighted-formula, lexicographic and Pareto.

The weighted-formula approach, also termed priori approach, is the simplest, easy to implement and it consists in assigning a weight to each objective, combining them in a single metric. Considering a quality metric Q, the different goals $g_1, g_2, \ldots, g_n$ and the assigned weights $w_1, w_2, \ldots, w_n$, the quality metric is often calculated using one of the following formulas (Cortez, 2021):

$$Q = w_1 \times g_1 + w_2 \times g_2 + \cdots + w_n \times g_n$$

$$Q = g_1^{w_1} \times g_2^{w_2} \times \cdots \times g_n^{w_n}$$

Although it is a simple solution, it has some issues associated with it. First, it is difficult to define the ideal weights to each objective, often requiring domain knowledge. Second, the weight definition task is often carried based on intuition, which negatively influences the result. Lastly, the fact that despite all objectives are being considered, only one variable is being optimized, often leads to the loss of possible trade-offs between them.

Similar to weighted-formula, the lexicographic approach gives a different importance to each goal. The difference between these approaches is that the latter ranks the objectives by priority and optimizes first the objective with higher priority, then the others. When two solutions are compared, if the first solution has a better metric value for the higher priority goal than the second solution, then the second is not considered despite the other goals' metric values. This approach has the same problems as weighted-formula concerning the priority rank to be defined a priori, however, the lexicographic approach does not mix non-commensurable criteria in the same formula (Cortez, 2021).

Lastly, the Pareto approach, when compared with the other approaches, brings more advantages and it is truly multi-objective, since it optimizes all objectives separately and simultaneously. This approach uses the concept of dominance, i.e., considering two different solutions, one dominates the other only if the first solution has a better metric value for one of the objectives and the remaining are either better or equal. Thus, the Pareto approach returns a set of non-dominated solutions (called Pareto's front) instead of a single one, allowing users to choose which one fits best their requirements. This type of approach does not need a priori weights or priorities to be defined and, since it deals with a population of solutions, EA have been a popular choice to generate Pareto optimal solutions. There are several *Multi-Objective Evolutionary Algorithm (MOEA)*, such as *Simple Elitist Evolutionary Algorithm (SEEA)*, *Strength Pareto Evolutionary Algorithm 2 (SPEA 2)*, yet the *Non-dominated Sorting Genetic Algorithm II (NSGA-II)* is the most widely used (Dhaenens and Jourdan, 2016) and it works as a classical GA with the particularity of handling multiple objectives with a Pareto approach.

2.5 Optimization Of Predictive Models

So far, several DM and Modern Optimization approaches and algorithms were addressed individually. Yet, this project aims to explore combinations of both, using Modern Optimization, in particular EA, to optimize DM models. The purpose is to increase the body of knowledge on this subject, by developing innovative methods, and use them to solve real-world complex problems, including a recent and interesting issue in the area of Mobile Performance Marketing (described in Section 2.6). Therefore, the current section describes current state-of-the-art approaches used on the modern optimization of predictive models topic, with a particular focus in EA used in two popular ML algorithms (one black-box and another white-box), namely NN and DT.

2.5.1 *Neuroevolution*

NN have been a target of great research, concerning the three types of learning (supervised, unsupervised and reinforcement learning), mostly due to advances and great achievements by Deep Learning in different subjects (e.g., anomaly detection, image classification, audio signal processing). Recently, there has been a particular interest on *Evolutionary Artificial Neural Network (EANN)*, which uses EAs for adjusting NN parameters. Similar to other DM models, NNs have a set of parameters that are typically fixed during its training phase, including the activation functions and the number of hidden layers and/or nodes in each layer (its topology), called hyperparameters. NNs also have a set of parameters that are adjusted during the training phase (weights and bias numeric values), that are traditionally adjusted using the backpropagation algorithm. These parameters can also be tuned using modern optimization algorithms, in particular, EAs. Neuroevolution is the subject that concerns with both NN topology (e.g., number of layers, nodes and connections between them) and weight and bias values optimization, either separately or simultaneously, using EAs.

Neuroevolution is a subfield of AI and ML whose goal is to trigger an evolutionary process similar to the human brain inside a computer. However, the human brain has an architecture with one quadrillion connections (Battiti and Brunato, 2017) that keeps changing along with time. Yet, in computer science, most of the times the network topology is fixed beforehand by the use, based on experience, and only the weight and bias values are adjusted during the network training phase. Usually, this phase corresponds to the optimization of these numerical values based on a predefined loss function, typically using backpropagation algorithm. However, there are two main disadvantages when using this approach: loss functions are quite inflexible and the optimization algorithm can be trapped on local minima, therefore harming the network learning phase. Nevertheless, both these issues can be tackled by using EAs instead of backpropagation, since these algorithms are global optimizers and their evaluation function is completely flexible and is defined by the user (see section 2.4.3). Moreover, when using multi-objective algorithms (e.g., NSGA-II), several goals can be considered (e.g., predictive performance and model's complexity).

Ojha et al. (2017) developed a review of 20 years of research concerning optimization of FNN using Modern Optimization algorithms, where the authors enumerate several implementations of EA for neuroevolution tasks. Concerning using fixed topologies, which is the more common approach, Ojha et al. stated that conventional algorithms, such as backpropagation, are outperformed by EA, mainly due to the lack of ability to escape local minima. There are several studies addressing the NN parameters optimization using Modern Optimization algorithms, for both single and multi-objective tasks, including Simulated Annealing (Budinich, 1996; Rere et al., 2015), PSO (Zhang et al., 2007), GA (Ruiz et al., 2018), NSGA-II (Furtuna et al., 2012) and, more recently, multi-objective DE (Kaur and Singh, 2021). Furthermore, these algorithms have been evaluated over different domains for supervised ML tasks, namely classification (Tirumala, 2020; Kaur and Singh, 2021), regression (Guijo-Rubio et al., 2020) and time-series forecasting (Mason et al., 2018). Additionally, several methods capable of simultaneously evolving both topology and weights of NNs were proposed, and this approach is designed as *Topology and Weight Evolving Artificial Neural Network (TWEANN)*. There are many implementations of TWEANN, including, but not limited to, *GeNeralized Acquisition of Re-current Link (GNARL)*, proposed in the 90s (Angeline et al., 1994), Cellular Encoding, also proposed in the 90s (Gruau, 1994), and *NeuroEvolution of Augmenting Topologies (NEAT)*, proposed in the 2000s (Stanley and Miikkulainen, 2002). Due to its popularity and effectiveness, NEAT was evaluated over several ML problems, most of them related to reinforcement learning tasks, and a few extensions emerged from the original algorithm.

NEAT is a GA-based algorithm that performs evolution of a FNN phenotype as a whole and applies special mutation and crossover operators in the network's nodes and connections (Ojha et al., 2017). According to Stanley (2017), part of NEAT's success relies on the ability of starting simply and increasing complexity over the generations. To do so, Stanley and Miikkulainen designed the algorithm to start with a simple SLP network, which means that it had only the input and output layers, and use mutation and crossover operators to increase the number of neurons, add connections between them and update their weight and bias values. Furthermore, Stanley states that this strategy used by the NEAT algorithm can end up creating a deep NN (Deep Learning) with hundreds of layers and neurons. In terms of its applications, most of them are reinforcement learning tasks, with a strong presence of videogame's automatic players (Kristo and Maulidevi, 2016; Pham et al., 2018). Regarding NEAT's extensions, several variants were proposed for playing videogames. For instance, *content-generating NeuroEvolution of Augmenting Topologies (cgNEAT)* was used in Galactic Arms Race (Hastings et al., 2009), *real-time NeuroEvolution of Augmenting Topologies (rtNEAT)* in the NERO videogame (Stanley et al., 2006) and in Open Racing Car Simulator (Cardamone et al., 2010), while its multi-objective extension termed *Modular Multi-objective NeuroEvolution of Augmenting Topologies (MMNEAT)* was used in Ms. Pac-Man (Schrum and Miikkulainen, 2014). Outside the domain of videogames, there is a successful extension also proposed by Stanley et al. (2009), termed *Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT)*, that can evolve high-dimensional NN and its weights are evolved as a function of geometry. Furthermore, with HyperNEAT, the authors were able to evolve a NN with hundreds of thousands to millions of connections (Stanley, 2017).

Even though TWEANN were very successful and their strength was proven several times in reinforcement learning tasks, this strength has not been yet demonstrated for supervised ML tasks. On the opposite, considering the particular case of strategic decision-making problems, both NEAT and HyperNEAT were not able to achieve the same success (Kohl and Miikkulainen, 2009; Lowell et al., 2011). Furthermore, HyperNEAT has been evaluated over the image classification problem and results show that it is better than others ML algorithms for feature selection, but was not successful as a classifier (Verbancsics and Harguess, 2015). Notwithstanding, a few studies have used both NEAT and HyperNEAT, and even an extension named Learning-NEAT, for classification problems and achieved interesting results (Chen et al., 2009; Pereira et al., 2019).

2.5.2 *Genetic Programming and Grammatical Evolution*

The previous section addressed the neuroevolution topic, where a particular focus was given to the creation and optimization of NNs using EAs. Nevertheless, modern optimization algorithms can also be used to design different ML algorithms (e.g., decision trees or symbolic regression). During this doctoral project we focused on two EAs in particular: Genetic Programming (Koza, 1990) and Grammatical Evolution (Ryan et al., 1998). The main reason for this selection lays on their domain-independent evolutionary process based either on a syxtax tree or on a user predefined grammar (Bianco et al., 2017), which allows flexibility to be applied to any issue. Both of these approaches have been used for different supervised machine learning tasks (Loveard and Ciesielski, 2006; Ashofteh et al., 2015; Fitzgerald et al., 2015), thus, the current section addresses this subject.

Regarding the use of GP, it typically comprehends the definition of a few operators with different arities. If the arity is zero, the value can, for example, come from attribute values and operators that have arities bigger than zero can combine these values, for instance using mathematical operators. For instance, Cortez (2021) presents how GP can be used to approximate mathematical functions, particularly the Rastrigin function by using the R programming language, with a few arithmetic operators: addiction, subtraction, multiplication and division. Concerning the use of GP for supervised learning tasks, two main approaches have been proposed (Iqbal et al., 2017): symbolic regression and classification. The former consists on the generation of mathematical functions able to map a set of inputs to the respective numeric output and it has been reported to achieve quite interesting results (Chen et al., 2019; Zhong et al., 2020; Ben Chaabene and Nehdi, 2021). On the other hand, several studies were proposed using GP as a classifier system, where its output can be a class or a class probability, by using GP trees as decision trees (Khoshgoftaar et al., 2003; Lee, 2006; Pradhan et al., 2012). Additionally, GP has also been applied to image classification tasks, being able to simultaneously apply feature selection, feature construction and class prediction, achieving competitive results when compared with other image classification algorithms (Bianco et al., 2017; Iqbal et al., 2017). Furthermore, different multi-objective extensions were proposed and evaluated under both regression (Ashofteh et al., 2015; Mehr and Nourani, 2017) and classification tasks (Zhao, 2007; Kessentini and Ouni, 2017). Due to its flexibility, GP can also be used for ML model optimization problems. For instance, Suganuma et al. (2017) used Cartesian GP with the purpose of defining Convolutional NNs topology and Zameer et al. (2017) used GP to build NN ensembles, combining their outputs.

In terms of GE, which is also a popular evolutionary computation technique (Bartoli et al., 2020), it is capable of evolving complete programs in any language, based on a user-defined context-free grammar (Ryan et al., 1998). In GE, genotypes, bit or integer strings, are mapped into the corresponding phenotypes, strings of a computer program, using a grammar in *Backus–Naur Form (BNF)*, which provides this technique a greater flexibility when compared with GP. It has been used in a wide range of problems (Bartoli et al., 2020), however, the current section focuses on machine learning tasks. Identically to GP, GE is also capable of evolving symbolic regression expressions (Sotelo-Figueroa et al., 2018; Bartoli et al., 2020; Nicolau and Agapitos, 2021). In particular, Nicolau and Agapitos (2021) performed a benchmark over 43 symbolic regression problems, using both GP and GE, aiming to find which function sets led to better generalisation performance, achieving quite interesting results. In terms of neuroevolution tasks, different studies have been proposed recently, using GE for both NN topology optimization (Quiroz-Ramírez et al., 2018; de Lima and Pozo, 2019) and TWEANN (Ahmadizar et al., 2015; Assunção et al., 2017). Specifically, Ahmadizar et al. claim to have obtained a good generalization ability in the GE-based networks. On the other hand, considering the grammar flexibility, GE has also been used for designing different classifiers (Fitzgerald et al., 2015; Nyathi and Pillay, 2018; Tzimourta et al., 2018) and has achieved competitive results when compared with multiple state of the art algorithms (Ahmadizar et al., 2015). Particularly, a special focus has been given to the creation and evolution of decision trees using GE in recent years (Fitzgerald et al., 2015; Rivera-López and Canul-Reich, 2018; Chabbouh et al., 2019; Czajkowski and Kretowski, 2019a). Such approaches consist in using GE to generate *if-then* rules, similar to DT, which is implemented by defining the grammar accordingly. Moreover, multi-objective approaches have been proposed recently (Chabbouh et al., 2019; Czajkowski and Kretowski, 2019b), considering either classification metrics and model complexity. The purpose of this approach is to achieve both high-quality and low-complexity decision tree classifiers. This subject is explored later on this document.

Regarding GA's evolutionary process, there are two additional interesting theories: Baldwin's and Lamarck's. Both of them combine GA with local-search methods. The former uses a local search during its evolutionary process, but changes on solutions are discarded, i.e., new solutions are not re-encoded nor kept during the remaining generations of algorithm's evolution. Thus, the local search is only used to improve the fitness of the evaluated solutions. The later, on the opposite, applies changes on solutions, be re-encoding the new solution and replacing the original one, in order to positively influence the evolution across the remaining generations. The Lamarckian approach was used in a neuroevolution problem presenting quite interesting results in (Cortez et al., 2002), supporting that this approach is quite interesting when performing a modern optimization of ML algorithms. In terms of comparison between both approaches, Rocha et al. (2004) used both, also in a neuroevolution task and compared results. According to the authors, the Baldwin effect presented greater improvements when comparing to a Lamarckian approach for their particular problem (Rocha et al., 2004). On the other hand, Mingo et al. (2013) performed the same comparison using an extension of GE combined with reinforcement learning, obtaining better results with the Lamarckian evolution. During this project, it is intended to address these GA hybrid approaches and compare to the original one, aiming to understand mains advantages and disadvantages.

Despite all mentioned advantages of using GP and GE, including the quite competitive results for ML tasks when compared with traditional algorithms, there is also a few disadvantages. An issue associated with both these methods is the code growth, often called *bloat* or *survival of the fattest* (Eiben and Smith, 2015). This phenomenon is observed when program trees tend to grow unnecessarily during the evolution process without corresponding to an increase in fitness, ending up generating non-functional code. Although the final solution is still effective and syntactically correct, it is not efficient since, in most cases, having more code leads to an increase on computational costs, and smaller code presents better generalization than longer code. Furthermore, according to Soule and Foster (1998), who studied bloat effects on GP, the code growth is associated with a slight preference for the larger of two programs created during the crossover. While defending that this overall growth does not affect the evolution of effective programs, they also state that non-functional code shielding functional code from the harmful effects of crossover may also decrease the chances of helpful crossovers to occur, leading to a stagnation on the GP evolutionary process (Soule and Foster, 1998).

Recently, several studies have emerged analysing the impact of bloat in both GP (Doerr et al., 2017; Juárez-Smith et al., 2019; Liang et al., 2020) and GE (Azad and Ryan, 2018; O'Neill and Brabazon, 2019; Hemberg et al., 2019). Such studies consider changes on the GA operators (mutation, crossover) or on the chosen arithmetic operators (in case of GP) or grammars (in case of GE). A way to escape the bloat problem is to repair the solution, removing the useless code, re-encode it and replace the bloated solution with the new one, so it can be passed to the following generations. This is a non-trivial task, since it involves automatically analysing the produced code (phenotype-level) and understand which pieces of it are redundant or useless, so it can be removed. A simpler way is to penalize solutions' fitness based on its size, benefiting, thus, smaller solutions. To do so, in some cases it might be necessary to estimate the minimum size necessary for a solution that is able to solve the problem, which also is not an easy task, for instance, when DT are being evolved. Yet, if a Pareto-based multi-objective algorithm is considered, the complexity of a solution can be used as a fitness metric, along with other metrics, and all used metrics can be optimized simultaneously. With this kind of approach, it is not necessary to estimate a minimum reasonable solution size, since the Pareto-based approaches will prioritize smaller solution sizes with a similar performance in the remaining metric (or metrics). During this doctoral project, it is intended to approach the bloat issue by using one of these last two approaches.

Another great disadvantage associated with both these methods is related to high computational costs (Suganuma et al., 2017; Fenton et al., 2017). Assuming a context where ML models are being evolved by GP or GE, the evaluation of solutions consists in using the designed models to perform predictions on a set of training data and measure its quality based on a predefined metric (e.g., *Area Under the Curve (AUC)*, *Mean Squared Error (MSE)*). Considering that both GP and GE are population-based optimization algorithms, which means that a set of solutions are evolved instead o a single one, this evaluation process can be quite expensive in terms of execution time. Moreover, these performance issues accentuate when larger sets of data are used. This doctoral project is associated with a complex real-world Big Data problem that includes, among others, time limitations (see Section 2.6). Therefore, during this project it will be necessary to deal with such issue.

The use of parallelism during the evaluation process is a simple, effective and popular way to deal with performance issues in GP and GA. Furthermore, this solution has been considered in the past for implementing GE in the Python programming language (Fenton et al., 2017). This parallelism approach consists in simply splitting the charge for multiple available cores, which in this specific problem implies sending a group of individuals to be evaluated simultaneously in different cores and, then, collect the results and continue with the evolution process. This approach, although it seems quite simple, may bring great advantages in terms of execution time and does not influence the fitness values. However, there is another parallelism approach used in GA that can bring great advantages not only in terms of computational costs, but also in terms of solution's fitness, named Island Model (Corcoran and Wainwright, 1994). The island model consists in having a set of cores evolving simultaneously their own population (islands) and periodically share the best solutions between the islands (migration), replacing the worst ones. With this approach, the concept of species is introduced, which refers to each population in each island. Since the individuals are evolved simultaneously and separately, in theory, each core is evolving a population with different characteristics, which increases the diversity of solutions. Island Model GA diversity was proven to lead to higher quality solutions when compared to the original GA (Corcoran and Wainwright, 1994; Whitley et al., 1998). In particular, Tsoulos et al.

(2008) proposed an Island Model GE-based model for creating and evolving NN, which was evaluated under 9 regression and 9 classification popular datasets and compared with 4 state-of-the-art methods for NN training. Quite interesting results were achieved, with the proposed method outperforming all other methods in all datasets, proving the strength of their method. During this PhD it is expected to explore and implement at least one of the mentioned parallelism alternatives to tackle the GA-based algorithms computational cost issue, thereby, fulfilling the project time requirements.

2.6 Mobile Performance Marketing

So far, several techniques were addressed regarding ML and Modern Optimization, both separately and combined, and some of their applications were also mentioned. Nevertheless, during this doctoral project we intend to address a specific problem in mobile performance marketing industry, often called performance-based advertising on mobile devices. Therefore, this section describe this problem, as well as how we intend to address it.

The mobile advertising industry is experiencing a considerable growth due to the great increase in mobile devices usage, being estimated at 100 billion dollars worldwide in 2016 (Du et al., 2016). Furthermore, according to Statista (2021), this industry has been target of great investments worldwide, that has been increasing over the last years as it can be seen in Figure 7. The mobile advertising business has 4 main stakeholders: users, publishers, advertisers and intermediary entities that own *Demand-Side Platforms (DSPs)*, such as OLAmobile, the company associated with this project. Publishers own popular digital spaces (e.g., news websites, online video games services) that attract a vast audience of users. Advertisers own marketing campaigns regarding products or services that they want to sell. Intermediary entities are responsible for linking publishers to advertisers using digital platforms (DSPs). In this market, compensation only occurs when there is a conversion, i.e., when a product or service is acquired by the user, or when mensurable metrics are presented. Common pricing models and metrics used in this kind of advertising include (Yuan et al., 2012; Du et al., 2016):

- ***Cost Per Mille (CPM)*** - price for showing 1000 impressions (advertisement displays) to potential customers;
- ***Cost Per Click (CPC)*** - price for each ad click, even though user clicks may not lead to a purchase;
- ***Cost Per Acquisition (CPA)/Cost Per Install (CPI)*** - price for any type of purchase or application installation after the click;
- ***Cost Per View (CPV)*** - price for every single ad display, usually in the form of pop-up or full-screen;

- ***Click Through Rate (CTR)*** - ratio between the number of ad clicks and the number of impressions;
- ***Conversion Rate (CVR)*** - percentage of clicks that originated a purchase over the total number of clicks;

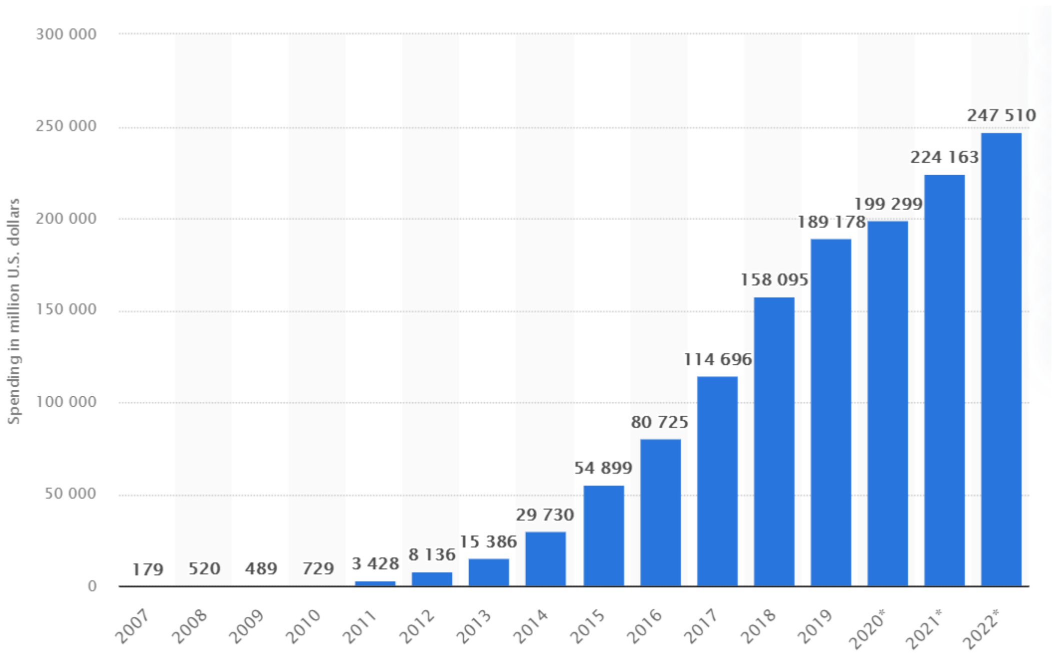


Figure 7: Investment on mobile advertising industry since 2007 [taken from (Statista, 2021)].

Compensation only occurs when conversions occur and the corresponding transaction is performed by advertisers to publishers and DSP owners. Publishers can be compensated by requiring users to click in a dynamic ad prior to accessing their contents. These ads are provided by the DSP, thus, their goal is to perform a good match between users and advertisement campaigns, aiming to increase conversions. Whenever a publicity campaign is presented and the user clicks on it, a dynamic link is activated and a redirection occurs, leading the user to the campaign page, so that the user can purchase the product or service. This redirection process is called redirect and is stored along with some user's information, such as the country where the access to that page happened and the user's device operative system. Whenever a redirect gives origin to a sale (conversion), the information is also stored with details about it, including the amount paid, the product or service acquired, the redirect that originated the sale, among other useful data attributes. Intermediary companies as OLAmobile intend to have a DSP that is able to automatically perform matches between users and campaigns, in order to increase users clicks and conversions and, consequently, increasing their revenues. There are two main types of conversions: sale and subscription. The former regards with product or service acquisition that lead to a single payment (e.g., songs, extra lives on video games), while the latter concerns with medium and long-term services with periodic payments (e.g., weekly news subscription). In case of subscriptions, advertisers concede a fixed value to the revenue, regardless of user's will to continue with the subscription. In these cases, an important term to consider is customer *Lifetime Value (LTV)*, which stands for the value of a customer considering possible future interactions that could be translated into benefit (Moro et al., 2015). Concerning the profit, both advertisers, publishers and intermediary organizations have greater benefits with long-term subscriptions. However, most of the purchases are single and often low-valued payments.

These markets are highly dependent of the DSP's performance, both in terms of choosing the right campaign and response time, since theses systems are responsible for matching users with useful ad campaigns within a 10 ms limit. Yet, according to OLAmobile business experts, nowadays DSP use generalist advertisements to present to users, based on the potential profit that publishers can achieve, which means that campaigns that generate higher revenue have higher possibility to be presented, although it may not fit to the user profile. Furthermore, currently this company has a very small conversion rate, with only approximately 1% of redirects originating sales. During this doctoral project, it is expected to use AI techniques to perform a match between users and campaigns, thus, increasing the DSP performance and, consequently, all involved entities' profits. Under this context, a key issue for the implementation of a DSP expert system is the design of a prediction model for the user CVR, which is often modeled as a binary classification task ("sale", "no sale"), aiming to estimate if a user will produce a conversion once a redirect occurred.

The mobile user CVR prediction task is nontrivial due to four main reasons (Matos et al., 2019a): it involves big data, with millions of redirects being generated every day; most redirects (e.g., 99%) do not result in conversions (it is a highly unbalanced task); only a limited set of input features is available (due to privacy and technology issues); and most features are categorical and contain a high cardinality (with hundreds or thousands of levels). Therefore, we intend to address the mentioned issues by using Modern Optimization algorithms for evolving ML models due to their success and flexibility, as presented in previous sections. Moreover, considering that different metrics are associated with this market, we also wish to explore multi-objective optimization algorithms. Following this rationale, the next section will present how similar problems were approached before, aiming to gain insights on the subject and find gaps in research that need to be fulfilled.

2.7 Optimization Of Predictive Models In Performance Marketing

Considering previous sections about the DM and EA algorithms, combinations of both and the mobile performance marketing problem, this section aims to present related studies and introduce a few problems that are intended to be addressed during this project. As previously mentioned, performance marketing industry deals with huge amounts of unbalanced data every day, particularly the organization under study deals with millions of records hourly. Traditional ML techniques have shown to be highly accurate in previous studies, however, dealing with such amounts of data brings other challenges beyond fitting such models to data. For instance, there are few traditional ML implementations that include parallel or distributed computation. Furthermore, most traditional algorithms are often evaluated under offline learning scenarios and, whenever online learning scenarios are considered, even if simply simulated, only a few models (e.g., deep learning) are capable of storing the knowledge obtained in previous training iterations and use it when fitting to new sets of data. The use of GA-based implementations allows to deal with such issues and, due to the evaluation function flexibility, it is possible to consider any possible metric or metrics (using multi-objective implementations), which is a great advantage over traditional ML algorithms.

Agarwal et al. (2014) presented a platform termed LASER, used for a social network advertising, using logistic regression and a set of techniques to ensure scalability and real-time responses. The purpose was to predict CTR for ads and their system was tested it in both offline and online learning environments. Moreover, the authors addressed the problem of having large amounts of unbalanced data on online advertisement industry, achieving good results (around 78% of AUC). To reinforce the importance of time limitations in terms of predictions, Agarwal et al. presented an approach based on the “better wrong than late” philosophy, explaining that is preferable to perform an incorrect prediction than not being able to predict in feasible time. Other studies addressed such challenges of dealing with mining high-speed data (Domingos and Hulten, 2000; Gaber et al., 2005; Krempl et al., 2014), yet, the unbalanced data issue and the performance marketing context were not considered. In 2015, Deng et al. developed a study where they proposed a framework for mobile advertising and marketing that was capable of suggesting an advertisement to a given user. The authors focused their study mainly in the Big Data component, developing only a pilot system where traditional ML algorithms such as decision trees and K-means are considered. Nevertheless, no specific metrics were presented and preliminary tests were performed using offline data and using only features concerning user’s location.

In a similar context, Moro et al. (2014) studied a bank telemarketing campaign that consisted in calling possible customers trying to sell a bank product. The study’s purpose was to predict the telemarketing call success, i.e., if the customer would, or not, purchase the product. Although it is a quite different industry from OLAmobile’s, and Modern optimization was not considered, the authors addressed an interesting goal, that is similar to the related studies on this subject (Du et al., 2016): predict user intention to purchase a product/service based on the user profile and call context. Furthermore, the same authors applied the customer LTV concept to the telemarketing campaign (Moro et al., 2015), which has shown to obtain increased advantages in terms of the predictive performance.

Recently, a preliminary study addressing the specific OLAmobile DSP was conducted by Du et al. (2016), where the authors intended to predict whether a given user’s intention to perform a purchase, once an campaign is displayed. Considering that only 1% of presented ads originate sales, this was considered a non-trivial ML task. In this study, authors used traditional ML algorithms, namely *Logistic Regression (LR)*, *Naive Bayes (NB)* and *Random Forest (RF)*, and were able to achieve very good results, measured in terms of AUC. During this doctoral project, other studies were conducted using OLAmobile data and addressing different issues (Matos et al., 2018, 2019a,b). In these studies, different data preprocessing strategies were employed, including data balancing (e.g., *Synthetic Minority Oversampling Technique (SMOTE)*) and label encoding techniques (e.g., *Inverse Document Frequency (IDF)*, One-Hot Encoding). Furthermore, other ML algorithms were used and compared, namely LR, RF, *XGboost (XB)*, OzaBoost, DecisionStump, Random Hoeffding Trees (Matos et al., 2018) and *Deep Learning (DL)* (Matos et al., 2019a,b). Nevertheless, none of these studies addressed the usage of a Modern Optimization to automatically design ML models.

The discussed research works have dealt with data streams, predictive models in mobile performance marketing or in similar marketing areas. Nonetheless, only traditional ML techniques were used and optimization methods, particularly metaheuristics, were not considered. Regarding this industry and even similar ones, previous to this PhD work, we did not find any studies that adopted Metaheuristics to design ML models. However, Trawinski (2013) used a GA to evolve an ensemble of fuzzy systems, which consisted in splitting the problem in smaller pieces and use one model for each piece. Their model was evaluated using data streams, related to a real estate market, comparing with other approaches, using statistical tests, and proving that the ensemble approach can be quite useful for complex online learning problems. Moreover, during the execution of this PhD project, Pereira et al. (2017) performed a preliminary study, also using the OLAmobile data, and that applied two neuroevolution state-of-the-art models, namely NEAT and HyperNEAT, comparing both with LR. NEAT implementation outperformed LR in terms of AUC, yet at the expense of a considerably higher computational cost.

Real-time biding is a popular approach associated with the mobile marketing industry that also addresses the problem of dealing with Big Data streams in a real-time environment. Considering that publishers are constantly renting their digital spaces, advertisers need to pay attention to what is happening by continuously monitoring publishers' metrics and prices, so that they do not lose a good business opportunity for displaying their campaigns. For instance, if an application or website is getting a sudden audience increase, it can be a good opportunity to rent that digital space there for presenting campaigns. These situations give origin to a real-time auction process, where the best bid gets the space for advertising. Du et al. (2017), using OLAmobile data, proposed a system, using a reinforcement learning approach, that used the number of clicks as a reward, i.e., the purpose was to select a digital space that would increase the number of clicks (redirects) on advertisements. In this study, Du et al. stated that their system outperformed the state-of-the-art bidding functions, concerning the number of clicks, for a low budget limit. Although this study addresses a mobile advertising industry problem associated with OLAmobile, it falls outside the scope of this doctoral project, which intends to explore Modern Optimization of ML models applied to supervised learning tasks.

2.8 Summary

The current chapter presented the main background knowledge of this thesis. During this doctoral project, we address a relevant issue in the mobile advertising industry where the purpose is to display interesting marketing campaigns to mobile devices' users, leading them to perform a purchase. This market has been gaining an increased attention and investment over the last years, boosted by the exponential increase of mobile devices usage (e.g., smartphones, tablets). In particular, we will use OLAmobile data, noting that it corresponds to a relevant company operating in this market that is responsible for matching users with relevant advertisements. The company deals with million hourly records related with displayed campaigns to mobile users, from which only a small percentage, nearly 1%, ends in conversions (sales). Considering the requirements associated with the Mobile Performance Marketing issue, we intend to address this issue by using Modern Optimization algorithms applied to ML models. In particular, we consider Evolutionary Algorithms, since they are easily adaptable to any problem and have been proven to be quite good in ML tasks. Therefore, we introduce the concepts of the three main topics during current chapter:

- **Big Data (section 2.2)** – although this project's main focus is associated with a ML task, the studied company deals with high-velocity and high-volume data, coming from different sources in a constantly changing environment, which is an inevitable technological issue to consider during this project. Thus, this section introduced useful concepts about this topic.
- **Data Mining (section 2.3)** – considering that this thesis explores the optimization of predictive models, during this section main DM and ML concepts are broadly introduced. To do so, different types of learning and a few popular learning algorithms are presented, with the focus being on the theoretical basis.
- **Modern Optimization (section 2.4)** – this section presents useful concepts related with the main focus of this thesis. Here, several optimization algorithms are introduced, from traditional to evolutionary ones, organized by the type of search. Furthermore, different multi-objective approaches are presented as well as possible algorithms to be considered.

After presenting the main theoretical concepts, the following ones concern with their applications in recent and related studies. The purpose is to understand how current state-of-the-art works applied the mentioned algorithms to identical ML tasks and business issues, and identify research opportunities. Therefore, the remaining sections of current chapter are organized as follows.

- **Optimization of Predictive Models (section 2.5)** – during this section, we focus on the use of EA for creating and evolving two particular ML models: NN and DT. The former, included in the Neuroevolution topic, presents different studies that optimize both the NN weights and architecture, either separately or simultaneously. From these studies it is possible to notice that

Neuroevolution is being the target of great interest, mostly due to the Deep Learning success, however, most of them are related to reinforcement learning tasks. Regarding DT, two popular and flexible EAs were addressed: GP and GE. In most of the studies, these methods were used to create symbolic regression expressions with only a few of them exploring the creation of white-box models like DT. Furthermore, the studies considering multi-objective optimization approaches are scarce, even though they can be quite interesting in the addressed context. Finally, we highlight the great computational cost associated with these algorithms and parallelism alternatives to tackle this issue.

- **Mobile Performance Marketing (section 2.6)** – this section details the performance marketing industry, with a particular focus in the company under study and its requirements. The increasing interest and consequent investments on this industry is highlighted here, as well as common pricing models and metrics used.

- **Optimization of Predictive Models in Performance Marketing (section 2.7)** – finally, this section presents some important works using ML techniques on the marketing industry, not limited to mobile devices. We present multiple studies, some of them developed in collaboration with OLAmobile by other fellow researchers along with this doctoral project. Nevertheless, we emphasize the lack of studies using EA for optimizing predictive models for supervised learning tasks, both in the mobile performing marketing industry and similar ones.

Throughout this chapter, several studies have been presented using ML algorithms to address different issues in marketing industries. However, to the best of our knowledge, there is not an existent approach that deals with optimization of these models, in particular by using Evolutionary Algorithms, in a constantly changing Big Data environment, that is able to deal with data unbalancement or in mobile marketing industry, nor in similar industries. Therefore, this PhD project intends to fill this gap, developing an optimization of predictive models and evaluating it using the OLAmobile DSP datasets. In particular, we intend to give particular focus to multi-objective approaches of EA, which can be quite useful to the mentioned company, and to the use of parallelism strategies, in order to fit the DSP temporal requirements.

Part II

MAIN BODY

3

MULTI-STEP TIME SERIES PREDICTION INTERVALS USING NEUROEVOLUTION

3.1 Introduction

Time-Series Forecasting (TSF), which models an event based on its past observations, is a crucial element to support decisions in several real-world domains, such as agriculture, economics, production, commerce and marketing (Makridakis et al., 1998). In particular, multi-step ahead TSF is useful for supporting tactical decisions, such as planning production resources or designing a marketing campaign several months in advance. TSF is usually modeled as a single point prediction task. However, *Prediction Intervals (PIs)*, set in terms of lower and upper bonds for the forecasts, are also invaluable in decision making, allowing to better estimate the uncertainty associated with key decision variables. For instance, a PI can be used to define what-if decision scenarios for the worst and best cases. As explained in (Khosravi et al., 2011), a PI includes more sources of uncertainty when compared with statistical confidence intervals, since it includes not only noise but also error measurements, lack of input data and even TSF model misspecification.

Due its importance, there is a vast scientific literature that addresses single point TSF, set in terms of two main approaches: Statistical and Soft Computing methods. Statistical methods are often developed within the field of Operations Research and are based on mathematical expressions (Chatfield, 2000). Popular examples include the Holt-Winters method and the ARIMA methodology (Makridakis et al., 1998). Soft Computing approaches are more related with the fields of Computer Science and include a range of distinct methods, such as (Stepnicka et al., 2013; Cortez and Donate, 2014): *Neural Networks (NNs)*, fuzzy techniques, *Support Vector Machines (SVMs)*, *Evolutionary Computation (EC)* and even hybrid combinations of the previous methods. In particular, several works have used EC to successfully optimize NNs for single point TSF (Peralta Donate and Cortez, 2014; Chandra and Chand, 2016), in a hybrid combination that is known as neuroevolution (Floreano et al., 2008).

This chapter addresses PI TSF using NNs, which is a much less researched topic and that involves two main measures of quality: PI coverage and width (Khosravi et al., 2011; Ak et al., 2013). The former is set in terms of number of point forecasts included in the PI, while the latter is defined by upper and lower bond overall difference. These measures often conflict. For instance, reducing a PI width tends to decrease the PI coverage. Thus, a trade-off needs to be set between the two measures. Prior to the year of 2011, several methods were proposed for regression PI using a NN as the base learner model. For example, a Bayesian method was used in 1992 to assign error bars to the NN predictions (MacKay, 1992). The method requires a high computational effort due to the calculation of derivatives and the Hessian matrix. The delta technique was proposed in 1996 (Chryssolouris et al., 1996). The technique contains two main limitations, it uses a linear NN model and it assumes that noise is normally distributed. The bootstrap is a more simpler approach that is more suited for small datasets (Heskes, 1996; Dybowski and Roberts, 2000). It assumes an ensemble of NN, each trained on bootstrap replicates of the training set. One limitation of bootstrap is that its adaption to the multi-step forecasting domain is non trivial, since there is a chronological order in the training series elements and boostrapping such elements would lead to sequential information loss. More importantly, as argued in (Khosravi et al., 2011), the Bayesian, delta and bootstrap methods contain two methodological drawbacks: the NN model was trained to optimize the single point prediction and not the PI; and the obtained intervals were only assessed using PI coverage but not width. In contrast, recent works (e.g., LUBE, LUBEX, MLUBE) assume a more natural approach where the PIs are directly optimized when fitting the NN and consider both coverage and with PI quality measures (Khosravi et al., 2011; Rana et al., 2013; Ak et al., 2013, 2015). In this chapter, we follow this natural approach, putting a stronger focus on two key state of the art PI models (LUBE and MLUBE) that were proposed for regression and that are extended for multi-step ahead TSF.

In 2011, the *Lower and Upper Bound Estimation (LUBE)* method was originally proposed for regression tasks (Khosravi et al., 2011). LUBE uses a *Multilayer Perceptron (MLP)* NN with two output nodes that correspond to the lower and upper PI values. LUBE optimizes a single and nondifferentiable objective function called coverage width-based criterion, which combines both coverage and width. Thus, a simulated annealing metaheuristic was used to train the NN weights instead of the conventional back-propagation algorithm. When compared with traditional PI methods (delta, Bayesian and bootstrap) in regression tasks, the LUBE method achieved competitive results. In 2013, an extension of the LUBE method was proposed for one-step ahead TSF PI prediction (Rana et al., 2013). The extension, named *Lower and Upper Bound Estimation eXtension (LUBEX)*, included a time lag feature selection and usage of an ensemble of NNs, where the upper and lower values were computed as the average of several individual NN outputs.

The first neuroevolution approach for PI was proposed in 2013, aiming also at regression tasks and defined by a direct encoding, where the weights of a fixed LUBE NN architecture were optimized using a *Multi-Objective Evolutionary Algorithm (MOEA)* (Ak et al., 2013). The proposed *Multi-objective evolutionary algorithm Lower and Upper Bound Estimation (MLUBE)* achieved interesting results in two tasks related with oil and gas deposition rates, although the model was not compared with LUBE. Nevertheless, MLUBE is more theoretically sound, since it evolves a Pareto front, thus simultaneously improving both coverage and width objectives, while LUBE only optimizes a single coverage/width trade-off. Later on, in 2015 (Ak et al., 2015), the same MOEA method was also adapted to fit NN to interval-valued time series data. Unlike LUBE, the base NN contained only an output node and the PI was obtained by feeding first the lower input values to the NN and then the upper inputs. Such as in LUBEX, only one-step ahead PI were considered. The proposed MOEA method compared favorably against an input-valued LUBE for wind speed data, although only a single run was executed and no statistical test was adopted.

In our preliminary work (Pereira et al., 2017), we extended both LUBE and MLUBE for multi-step TSF PIs and performed comparative experiments over four times series. The comparison comprised a more robust evaluation methodology that included the realistic rolling window procedure (Tashman, 2000) and the Wilcoxon nonparametic statistical test (Hollander et al., 2013). In this chapter, we present a more comprehensive set of PI TSF methods, putting an emphasis on neuroevolution approaches. The set of PI methods include: adaptations of LUBE, LUBEX and MLUBE for multi-step TSF (LUBET, LUBEXT and MLUBET), a two phase learning MLUBE (M2LUBET) and new ensemble neuroevolution versions (MLUBEXT, M2LUBEXT). We also perform a wider comparative study, using a robust evaluation and nine time series from distinct real-world domains (e.g., Agriculture, Retail, Tourism).

The current chapter is organized as follows. Subsection 3.2.1 presents first the time series data. Then, the forecasting methods are described in Subsection 3.2.2. In particular, single-point NN TSF is briefly presented in Subsection 3.2.2. To better describe all PI adaptations and aiming at self-containment, Subsection 3.2.2 first details the LUBE original model and then its LUBET and LUBEXT extensions. Similarly, Subsection 3.2.2 introduces first MLUBE and then its time series and ensemble variants: MLUBET, M2LUBET, MLUBEXT, M2LUBEXT, MLUBEXT2 and M2LUBEXT2. Next, Subsection 3.2.3 explains the evaluation procedure. Subsection 3.3 details the performed experiments and obtained results, with both single NN and ensemble NN methods. Finally, Subsection 3.4 summarizes the work developed and withdraws the main conclusions.

3.2 Materials And Methods

3.2.1 *Time series datasets*

This work addresses seasonal series, since multi-step forecasts are particularly relevant for these types of cycle patterns, which are expected to reoccur in the future. Moreover, seasonality is common in several real-world domains (e.g., monthly sales or quarterly production numbers) (Makridakis et al., 1998). Table 1 presents the main characteristics of the nine time series that were selected from distinct domains and with different characteristics. As shown in Figure 8, in addition to the seasonal component the datasets exhibit distinct trend components: some present a growth trend (e.g., gas, pass, water), some are more stationary (e.g, MG, suns) and crad presents a changing trend effect.

Seven series (cradfq, gas, pass, pigs, suns, usauto, water) were retrieved from the well known Time Series Data Library (TSDL) public repository (Hyndman, 2010), while MG is a chaotic series (Glass and Mackey, 1977) and store was collected and detailed in (Cortez et al., 2016). Except for MG, all series are from real-world domains. As argued in (Peralta Donate and Cortez, 2014), such real-world datasets are often affected by external phenomena, such as economical cycles or meteorological conditions, making them interesting series that are more difficult to predict.

Table 1: Description of the selected time series datasets (L – series length, K – seasonal period, W – rolling window size, S – rolling window step).

Series	**Description (location; years)**	L	K	W	S
cradfq	Monthly highest radio broadcasting freq. (Washington, D.C., USA; 1934-1954)	240	12	199	1
gas	Monthly gasoline demand, in gallon millions (Ontario, Canada; 1960 – 1975)	192	12	151	1
MG	Mackey-Glass synthetic chaotic series (–;–)	783	17	505	9
pass	Monthly airline passengers, in thousands (–; 1949-1960)	144	12	103	1
pigs	Monthly number of pigs slaughtered (Victoria, Australia; 1980-1995)	188	12	147	1
store	Daily number of customers that entered a sports store (Portugal; 2013)	257	7	221	1
suns	Yearly number of sunspots (–; 1700-1988)	289	10	220	2
usauto	Monthly auto registration numbers, in thousands (USA; 1947-1968)	264	12	193	2
water	Monthly water usage, in ml/day (London, Ontario, Canada; 1966-1988)	276	12	205	2

3.2.2 *Forecasting methods*

Single point forecasting

Time series consists of several time ordered values related with the same event: $y_1, y_2, ..., y_L$, where L is the length of the series. An autoregressive forecasting model produces the estimate $\hat{y}_{t+1}$:

$$\hat{y}_{t+1} = f(y_{t-k_I+1}, ..., y_{t-k_1+1}) \tag{1}$$

where f is the forecasting function (e.g., linear regression, NN), t is the current time (associated with the last known value y_t) and k_i denotes a time lag. Often, a sliding time window with $\{k_1, ..., k_I\}$ time lags is used to create supervised training cases to fit the learning methods (e.g., MLP, support vector machine), thus using a soft computing approach to define the f function. For example, if the $\{1, 3\}$ window is used for the series $5_1, 11_2, 15_3, 17_4, 22_5$ (y_t values), then two training cases can be generated: (5,15) → 17 and (11,17) → 22.

The fully connected MLP is a popular NN for single point TSF (Cortez et al., 2012; Stepnicka et al., 2013; Peralta Donate and Cortez, 2014). The regression model is often set in terms of: an input layer

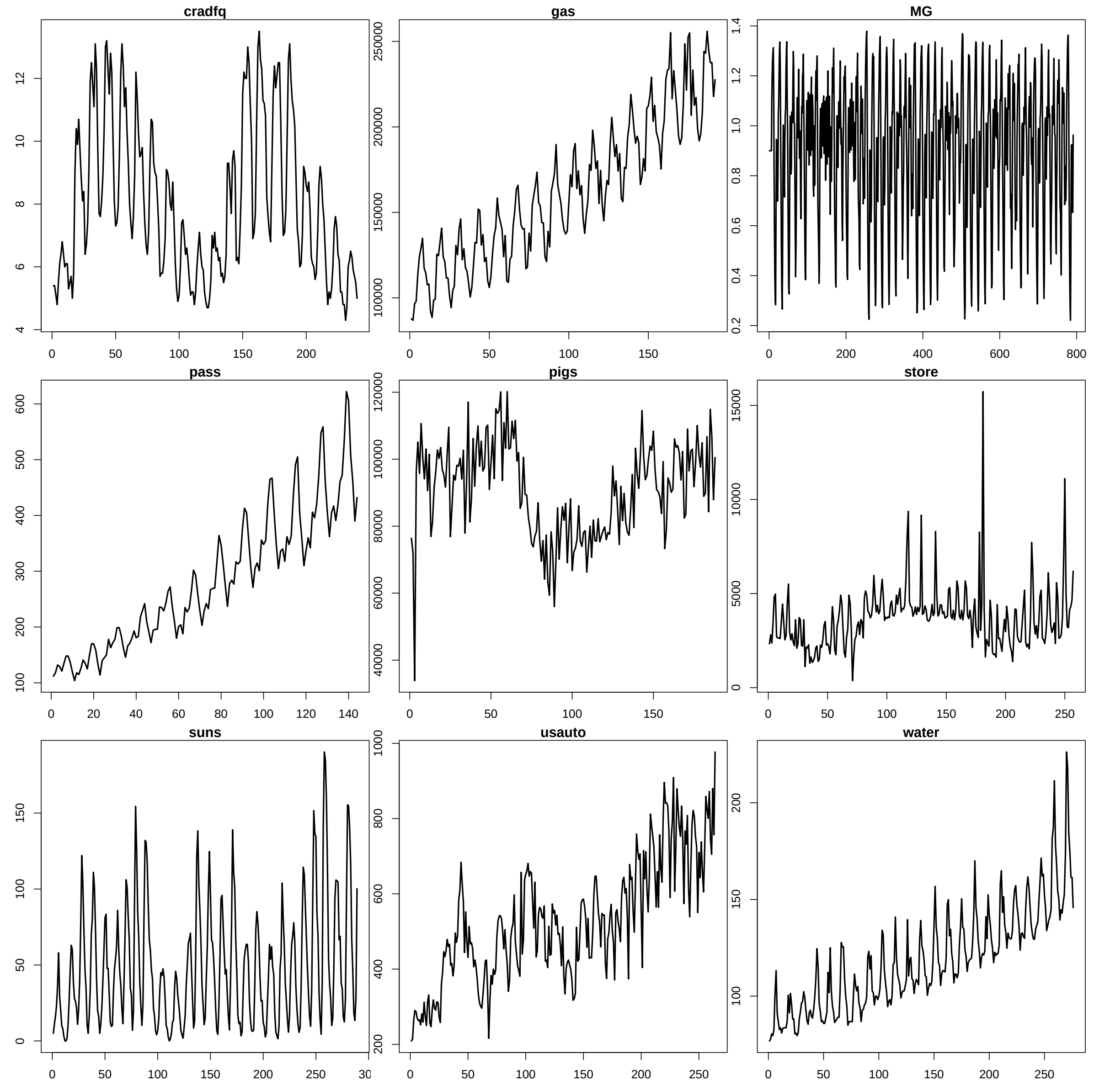


Figure 8: Time series plots (x-axis denotes the time period t, y-axis the time series y_t value).

with I inputs, the sliding window time lags; a hidden layer with H hidden nodes and logistic activation functions; and an output layer with one output linear function node. The forecasts are given by:

$$\hat{y}_{t+1} = w_{2:0,1} + \sum_{j=1}^{H} S(w_{1:0,j} + \sum_{i=1}^{I} y_{t-k_i} w_{1:i,j}) \times w_{2:j,1} \tag{2}$$

where $w_{m:i,j}$ denotes the weight connection with from layer $m-1$ and node i to layer m and node j (if $i=0$ then it is a bias connection) and S the logistic (sigmoid) function: $S(x) = \frac{1}{1+e^{-x}}$. Figure 9 shows several examples of $w_{m:i,j}$ weight connections.

Ensembles can be used to combine predictions from several forecasting models. Often, ensembles achieve better performances when compared with their individual prediction models (Dietterich, 2000;

Yu et al., 2017). In particular, ensemble averaging is a simple and popular approach to combine regression outputs from several learning models (Oliveira et al., 2017). For example, such ensembles have been used to combine single point time series predictions from distinct MLPs (Cortez et al., 2012).

To perform multi-step forecasting, several predictions are made at time t, from $\hat{y}_{t+1}$ (1-ahead) to $\hat{y}_{t+h}$ (h-ahead, where h is the forecasting horizon). A popular method to achieve multi-step predictions is to use iterative feedback (Peralta Donate and Cortez, 2014). Under this method, the model produces first an 1-ahead forecast. Then, this forecast is used as the last input of the forecasting function to generate the 2-ahead prediction, and so on.

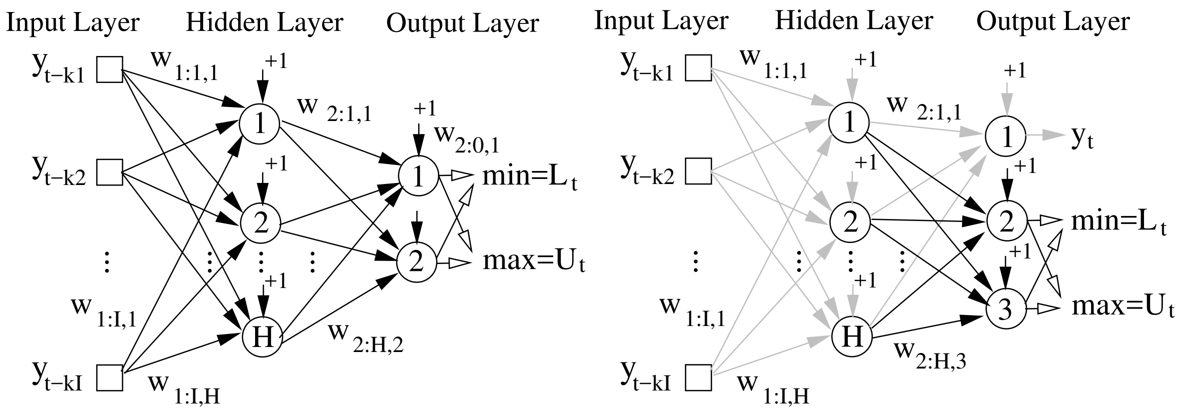


Figure 9: The base Neural Network model for LUBE (left) and M2LUBET (right, first phase fixed weights in gray color).

Single objective prediction interval methods

This subsection starts by detailing the LUBE model, which is a fundamental NN structure used by all PI methods explored in this chapter. LUBE (Khosravi et al., 2011) was the first proposed method to directly optimize PIs. It uses a base learner that is similar to the single point TSF MLP (subsection 3.2.2), except that the output layer now contains two linear nodes with the lower (L_t) and upper (U_t) prediction interval (PI) bounds, as shown in the left of Figure 9. The number of weights that are optimized is

$$\#w_{\text{LUBE}} = (1 + I) \times H + (1 + H) \times 2 \tag{3}$$

where the $+1$ term is due to the use of bias weights ($w_{m:0,j}$), # is the cardinality operator and w_{M} is the vector that contains all fitted weights ($w_{m:i,j}$) for method M. The NN is trained using a simulated annealing that minimizes the *Coverage Width-based Criterion (CWC)*:

$$
\begin{aligned}
c_i &= \begin{cases} 1 & \text{if } y_i \in [L_i, U_i] \\ 0 & \text{else} \end{cases} \\
PICP &= \tfrac{1}{n} \textstyle\sum_{i=1}^{n} c_i \\
NMPIW &= \tfrac{1}{R \times n} \textstyle\sum_{i=1}^{n} U_i - L_i \\
CWC &= NMPIW(1 + \gamma(PICP) e^{-\eta(PICP-\mu)})
\end{aligned} \tag{4}
$$

assuming the prediction interval (PI) of $[L_i, U_i]$ for time period i, where L_i and U_i denote the lower and upper bounds, c_i is the coverage function, n is the number of PI estimates, $PICP$ is the *Prediction Interval Coverage Probability (PICP)*, $NMPIW$ is the *Normalized Mean Prediction Interval Width (NMPIW)*, R is the range of the target, η and μ are constants, and γ is confidence level step function ($\gamma = 0$ if PICP $\geq \mu$; otherwise $\gamma = 1$). When PICP is higher than the confidence level (μ), CWC produces a small value that is equal to the NMPIW, otherwise the PICP is considered unsatisfactory and CWC increases exponentially due to the value of γ.

LUBEX is an extension of LUBE that adopts a time lag feature selection and an ensemble averaging (Rana et al., 2013). The latter feature works by first storing the best NR solutions when executing the simulated annealing optimization of LUBE. Then, the PI ensemble is produced by averaging the upper and lower estimates of the distinct NR models.

The μ and η constants are hyperparameters that define how much penalty is given to PI with a low coverage. In (Khosravi et al., 2011), authors assumed the values $\mu = 0.90$ and $\eta = 50$ that highly penalize PI with a coverage probability lower than 90%. Our claim is that optimizing a single measure, such as CWC, is a more limited approach when compared with the multi-objective neuroevolutionary approaches (subsection 3.2.2), since it ignores distinct coverage-width trade-offs. Given that we need to fix *a priori* the μ and η constants for optimizing LUBEX, in this work we assume the default $\eta = 50$ and $\mu = 0.90$ values proposed in (Khosravi et al., 2011). To achieve a fair comparison, both LUBE and LUBEX are trained with the same inputs (generated by the sliding window with I time lags). To set the best number of hidden nodes (H), the training data is first split into training and validation sets. Then, a grid search is used to explore distinct H values and select the minimum CWC validation network. After setting H, the NN is retrained with all training data.

To facilitate the metaheuristic optimization, we do not assign fixed roles to the two output nodes. Instead, the lower and upper values are automatically set as the minimum and maximum of the output vales. For example, consider a NN with a base structure similar to the left of Figure 9. For a particular time series input window, the NN could return the pair of values (230 – first output node, 137 – second output node). If the NN output node roles were previously fixed to the lower (first output) and upper (second output) then the interval would be infeasible, with $U_t < L_t$. In contrast, the flexible minimum and maximum role assignment leads to a feasible interval: $[\min(230, 137) = 137, \max(230, 137) = 230]$.

LUBE was originally proposed for regression tasks, while LUBEX only handled one-step ahead TSF. In previous work (Pereira et al., 2017), we adapted both methods to perform multi-step ahead TSF PI. The adaptation, termed here LUBET and LUBEXT, assumes computing the next ahead estimate as the middle of the PI: $\hat{y}_{t+i} = (U_{t+i} + L_{t+i})/2$, where $i \in 1, ..., h$ and L_{t+i} and U_{t+i} represent the predicted lower and upper bounds for time $t + i$. Similarly to the single point multi-step TSF, an iterative feedback of the $\hat{y}_{t+i}$ values is used to generate the i-ahead PI and middle values.

Neuroevolution prediction interval methods

In this subsection, the evolutionary multi-objective method (MLUBE) (Ak et al., 2013) is first presented, since it is the base model for all proposed neuroevolution TSF PI methods. MLUBE was originally proposed for regression tasks and it uses the same fixed LUBE NN architecture and a direct real-valued representation chromosome with a total of $\#w_{\text{MLUBE}} = \#w_{\text{LUBE}}$ (Equation 3) weights. The weights are set by using the *Non-dominated Sorting Genetic Algorithm II (NSGA-II)* (Srinivas and Deb, 1994) to evolve a Pareto front that contains all nondominated solutions, i.e., the best multi-objective trade-offs. We adopt the NSGA-II algorithm for the multi-objective optimization because it was used in (Ak et al., 2013, 2015) and we wish to have a fair experimental comparison, thus using the same algorithm to compare the distinct MLUBE based methods (MLUBET, M2LUBET, MLUBEXT and M2LUBEXT).

The NSGA-II algorithm optimizes a population with a size of P_s individuals that are initially set randomly, where each individual (or solution) is composed of a vector of real values (the NN weights). In each new generation, new individuals are created by using two genetic operators (Srinivas and Deb, 1994): a single-point crossover, with a crossover probability of Pc; and polynomial mutation, with a mutation probability of P_m. Then, all individuals are evaluated, using two fitness functions: *Prediction Interval Coverage Error (PICE)* $= 1 - PICP$ (the PICE) and $NMPIW$ (the PI width). The NSGA-II algorithm selects for the next generation the best fitted individuals by using a non-dominated Pareto sorting approach and that considers both fitness functions. This allows the algorithm to evolve iteratively a Pareto front, until it is stopped after G generations.

The LUBE method uses an internal validation procedure to select the best number of hidden nodes (H) based on the single objective CWC metric. MLUBE also uses a similar H value procedure but with an adapted selection criterion. Since MLUBE simultaneously evolves two objectives ($PICE$ and $NMPIW$), instead of CWC we use the highest hypervolume (Beume et al., 2009) selection criterion, which correlates with interesting Pareto fronts. In effect, the hypervolume is a popular measure used to compare Pareto fronts generated from distinct optimizations and it is computed by considering a reference point. Figure 10 shows an example Pareto front. Often, the reference point is set as the worst solution. In this work, we assume such worst point reference as 100% PICE and 100% NMPIW, i.e., $(1, 1)$. Thus, the ideal hypervolume value is 1.0. Since MLUBE was originally only proposed for regression tasks, we also adapted this method for multi-step PI (denoted by the term MLUBET), where the middle of the PI is used to provide iterative feedback values ($\hat{y}_{t+i}$) (Pereira et al., 2017).

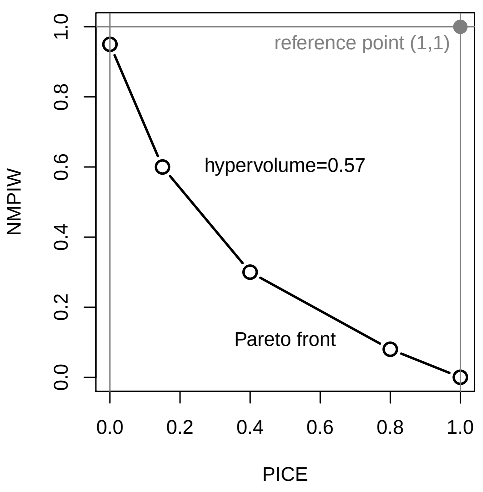


Figure 10: Example of a Pareto front with five PI coverage and width trade-offs (white circles) and its hypervolume value when using the (1,1) reference point.

.

In our previous work (Pereira et al., 2017), we introduced two variants for MLUBET, both based on a 2-phase learning (M2LUBET). The first phase involves a backpropagation training of a single point TSF MLP, with one output node (subsection 3.2.2). Then, the obtained weights are fixed (gray connections shown at the right of Figure 9). In the second phase, two additional output nodes, and their respective connection weights, are added to the model, in order to allow the estimation of the lower and upper values. Next, the new second phase weights ($w_{2:i,j}, j \in \{2, 3\}$) are optimized using NSGA-II (black connection weights shown at the right of Figure 9). Thus, the multi-objective optimization is performed over a smaller search space when compared with MLUBET, since $\#w_{\text{M2LUBET}} = (1 + H) \times 2$.

The difference between the two variants (M2LUBET1 and M2LUBET2) is related with the generation of multi-step PI. M2LUBET1 uses the first output node to directly generate the $\hat{y}_{t+i}$ estimates, while M2LUBET2 uses the middle PI point (average of the second and third outputs). In this work, we only consider M2LUBET2 as the M2LUBET representative, since it adopts the same multi-step approach used by other methods (e.g., LUBET, MLUBET) and it provided better results in the preliminary experiments conducted in (Pereira et al., 2017).

The ensemble averaging adopted by LUBEX provided better PI results in terms of the single CWC criterion (Rana et al., 2013). Inspired in this result, we propose new ensemble versions for neuroevolution PI methods. These ensemble adaptations are more complex than LUBEX, since multi-objective methods optimize a Pareto curve, with distinct coverage-width trade-offs. Thus, KC ensembles need to be built, each one related with a distinct optimization region. To achieve this, we first store all Pareto curve solutions achieved during the NSGA-II evolution. Then, we split the optimized coverage-width space into KC zones according to two main strategies: radial slices and clustering. The former strategy considers all evolved Pareto curve points, dividing them into KC equal sized radial slices (left of Figure 11). The latter approach clusters the most recent Pareto curve points (from the last LG generations, which tend to include points closer to the best Pareto front) into KC groups using the k-means algorithm (Witten et al., 2011) (right of Figure 11). Next, we select the best set of NR MLPs for each of the KC zones. In both strategies, the selected NR models are the ones that are farther away from the (1,1) reference point, using the Euclidean distance, in coverage-width space. Finally, the PIs of these NR models are then averaged (as in LUBEX), in order to compute the ensemble PI. In this chapter, the radial slice ensembles for MLUBET and M2LUBET are termed MLUBEXT and M2LUBEXT, while the clustering ensembles are denoted as MLUBEXT2 and M2LUBEXT2.

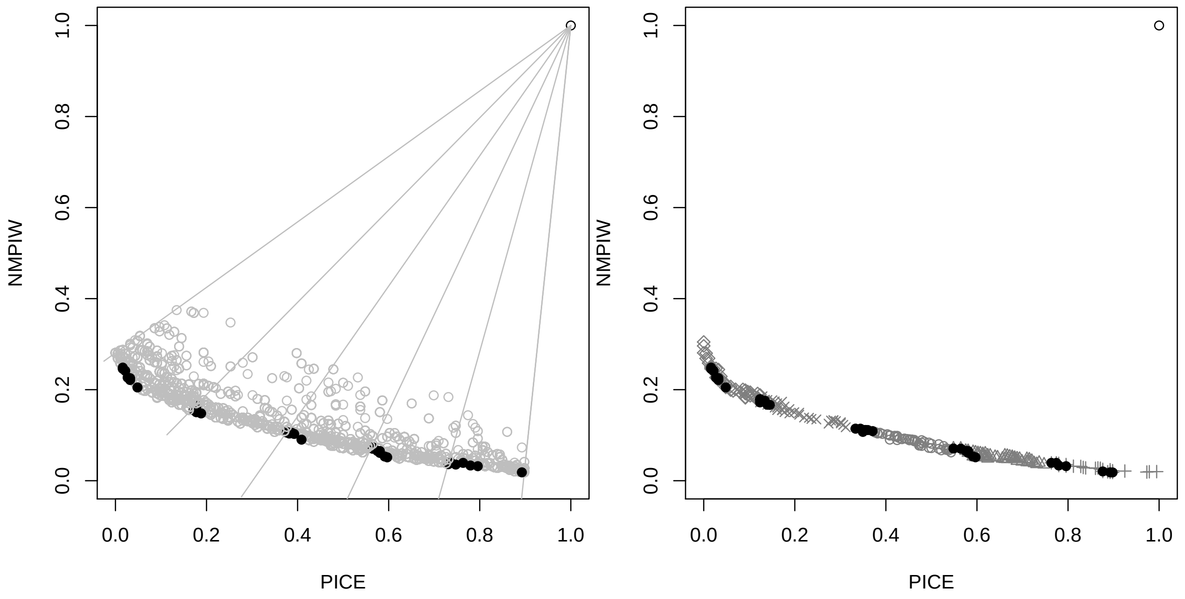


Figure 11: Example of the radial slices (left) and clustering (right) strategies for ensemble model selection ($KC = 5$ and $NR = 7$; the analyzed Pareto points are in gray; the left graph ensemble regions are separated by radial slices, while each right graph cluster is denoted by a distinct symbol; the selected NR models are in black).

3.2.3 *Evaluation*

Forecasting methods are often compared using a time ordered holdout, where the data is split into training and test elements (Cortez et al., 2012). In this work, we adopt the more robust and realistic rolling window estimation (Tashman, 2000), which allows the generation of several training and testing iterations. The rolling window contains three main parameters (Figure 12): W – the size of the training window; h – the number of multiple step ahead predictions; and S – the step size, i.e., the number of window elements are updated in each new iteration. In the first iteration, the training set is composed of the oldest W elements of the time series and it is used to fit the PI method, which estimates, at time $t = W$, 1 to h multistep ahead PI. Then, the PICE and NMPIW forecasting measures are computed using the time series test samples from time $t + 1$ to $t + h$. In the next iteration, the training window is updated by deleting its oldest S values and adding the more recent $t + S$ time series observations. The forecasting model is then retrained and at time $t = W + S$ new h multi-step ahead forecasts and performance measures are computed, and so on. In total, the rolling window produces $U = (L - (W + h - 1))/S$ iterations.

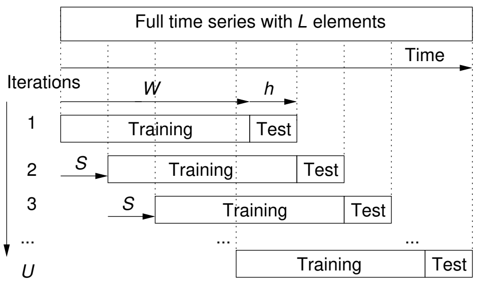


Figure 12: Schematic of the rolling window procedure.

In order to achieve a more fair comparison, the same inputs are used for all tested PI methods, assuming a sliding time window with all time lags up to $I = K + 1$ ($k_i \in \{1, 2, ..., K, K + 1\}$), which allows the inclusion of the seasonal pattern (K) plus a possible trend (Stepnicka et al., 2013). Table 1 presents the K values for the selected time series.

The number of MLP hidden nodes (H) is fixed for each PI base model (LUBET, MLUBET or M2LUBET). To reduce the computational effort, H is only optimized during a preprocessing stage that is performed before the rolling window execution. Using the first training data (oldest W time series observations), the optimization uses a grid search where several H values are tested by splitting the data into fitting (oldest 70% elements) and validation (recent 30% values) sets. Given that the MLP training is stochastic, a total of R runs are executed for each H value. Then, the H value is selected, by considering the best average CWC (LUBET, LUBEXT) or hypervolume (other methods) validation measures, and the normal rolling window procedure is executed.

To compare the PI methods, the test set results need to be aggregated, since there are U test sets. For the multi-objective methods, this results in U distinct Pareto fronts. Moreover, some non-dominated solutions in the training sets can correspond to dominated points in test sets. As such, the full PI test results often contain several width values for the same coverage, as shown in Figure 13. Thus, the statistical comparison of PI methods is non trivial. Inspired by *Receiver Operating Characteristic (ROC)* curve vertical aggregation (Fawcett, 2006), we propose a similar procedure to compare the PI results from all U iterations. The results are first aggregated vertically, where for each different PICE value the NMPIW median and respective 95% confidence intervals are estimated, according to the nonparametric Wilcoxon test (Hollander et al., 2013). An example of such median curve is shown in Figure 13. Finally, the overall hypervolume is computed using the estimated Pareto median curve and the adopted (1,1) reference point.

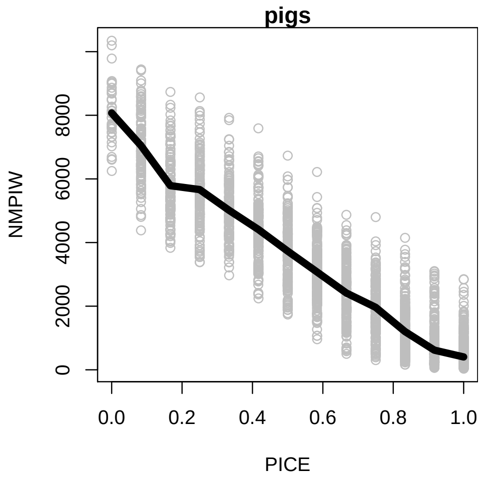


Figure 13: Example of the forecasting results for MLUBET using the pigs dataset (full test results are in gray; the estimated median curve is in black).

3.3 Results

3.3.1 *Experimental setup*

The experiments were carried out using the R computational environment (R Core Team, 2016) and run on a dedicated Linux Intel Xeon 1.7 GHZ server. In particular, the MLP backpropagation, simulated annealing and NSGA-II fit was performed using the `nnet`, `optim` and `nsga2` functions from the `nnet`[1], `stats`[2] and `mco`[3] R packages.

In forecasting experimental comparisons, it is quite common to have several modeling parameters. Since its computationally unfeasible to test all parameter combinations, some values need to be fixed using reasonable assumptions (Hand, 2006). In this work, we adopted the setup that is summarized in Table 2 and that is detailed in the next paragraphs. To compare the methods in a fair way, we used the same evaluation procedure for all PI methods (rolling window parameters, same number of inputs). Furthermore, when possible, we adopted the default algorithm parameters, as suggested in the state of the art works (e.g., Khosravi et al. (2011); Rana et al. (2013)) or when implemented in the R tool. Similarly to Khosravi et al. (2011), the number of NN hidden nodes (H) is set by using an internal grid search while the ensemble parameters (NR, KC, LG) were set using preliminary experiments.

The R environment default parameter values include: crossover probability of P_c=0.7, mutation probability of P_m=0.2, population size of P_s=100 and maximum of G=100 generations for NSGA-II; and 100 epochs of the BFGS backpropagation algorithm used in the first of the two-phase learning methods. The simulated annealing was set with the values adopted in Khosravi et al. (2011); Rana

1 https://cran.r-project.org/web/packages/nnet/index.html
2 https://rdocumentation.org/packages/stats/versions/3.6.2
3 https://cran.r-project.org/web/packages/mco/index.html

et al. (2013) (initial temperature of 5.0 and $\eta = 50$ and $\mu = 0.90$) and it was stopped after $G = 100 \times 100 = 10,000$ iterations, which corresponds to the same level of NSGA-II searched solutions.

Preliminary experiments were held to test several numbers of multilayer perceptrons (e.g., $NR \in \{5, 7, 9, 11\}$) for the ensemble methods and using only the first rolling window training data for series cradfq and MG. Better validation results were achieved for the value $NR = 7$, which was kept fixed for all ensemble methods and time series. In case of the neuroevolution ensemble methods, the coverage-width space was divided into $KC = 25$ regions, which allows to define a large number of trade-offs. We also explored slight different values (e.g., $KC = 23$, $KC = 27$) in preliminary experiments but no substantial differences were achieved and thus we kept the initial $KC = 25$ setup. For the clustering ensemble partition method, LG was set to the last 25 generations.

Table 2: Summary of the main parameters used in this study.

Context	**Parameter setup**
Time series	seasonal period K is specific to each series (Table 1)
Rolling window	W and S were adjusted (Table 1) to achieve U =30 iterations The horizon h is set equal to the seasonal period K
Inputs	The number of inputs is $I = K + 1$ Stepnicka et al. for LUBE and MLUBE
LUBE	Simulated annealing with initial temperature of 5.0 Simulated annealing stopped after 10,000 iterations Evaluation function of CWC with $\eta = 50$, $\mu = 0.9$ Khosravi et al.; Rana et al. Hidden nodes set by internal validation with CWC criterion ($H \in \{0, ..., 9\}$)
MLUBE	NSGA-II with $P_s = 100$, $P_c = 0.7$, $P_m = 0.2$ NSGA-II stopped after G =100 generations Two evaluation functions: $PICE$ and $NMPIW$ Hidden nodes set by internal validation with hypervolume ($H \in \{0, ..., 9\}$)
Ensembles	$NR = 7$ prediction models and $KC = 25$ regions (preliminary experiments) most recent $LG = 25$ Pareto fronts used by the clustering ensembles

The training data was first standardized to a zero mean and one standard deviation (Hastie et al., 2009). Also, all initial MLP weights were randomly set within the range $[-1, 1]$. In case of NSGA-II, this means that the lower and upper gene bounds were also set within this range. After training the models, the MLP outputs were post-processed with the inverse of the standardized function.

The rolling window evaluation procedure was defined with a reasonable large number of iterations ($U = 30$) and its window and step size parameters (W and S) were adjusted to the time series length, as presented in Table 1. The forecasting horizon was set equal to the seasonal cycle ($h = K$). For instance, the next 12 PIs are estimated for the monthly series, which corresponds to a full year prediction.

Regarding the search of best hidden nodes (H), a total of ten searches were performed using the training data of the first rolling window iteration, in a grid search that ranged from $H = 0$ (simpler linear regression) to $H = 9$ (more complex MLP). Each hidden node configuration was trained $R = 10$ times and the average validation estimation measure (CWC or hypervolume) was used to select the best configuration. A grid search was performed for each different base learner, resulting in the three hidden node selections (for LUBET, MLUBET and M2LUBET based models) presented in Table 3. For all series, the single-objective models (LUBET) favor small networks, with just one hidden node. In contrast, larger models are chosen when using the two neuroevolution base models (MLUBET and M2LUBET), ranging from $H = 4$ to $H = 9$.

Table 3: Selected number of hidden nodes (H) for the distinct base learner models.

Series	**LUBET**	**MLUBET**	**M2LUBET**
cradfq	1	7	6
gas	1	6	7
MG	1	8	9
pass	1	5	8
pigs	1	6	5
store	1	5	7
suns	1	7	9
usauto	1	7	6
water	1	5	4

3.3.2 *Forecasting methods*

Since we compare a large number of PI methods, we first analyze the non-ensemble methods and then the ensemble based ones. Figure 14 shows the non-ensemble method results in terms of the Wilcoxon estimated median Pareto curve and its respective 95% confidence intervals. The coverage-width forecasting graphs clearly show that LUBET is outperformed by both MLUBET and M2LUBET methods for all time series, even in the low coverage error (PICE) region, which is the area favored by the CWC criterion. In most cases (e.g., cradfq, gas, mg, pass, gas, pass, usauto, water), the neuroevolution results are considerably better, with higher than 0.2 point differences when compared with LUBET in terms of NMPIW for the same PICE value. Moreover, the 95% confidence intervals do not overlap, showing statistically significant differences. Regarding the neuroevolution method comparison, the performances are more similar, presenting smaller differences that depend on the dataset and PICE region analyzed. For instance, M2LUBET shows a substantial improvement over MLUBET for a PICE value around 0.1 for the water series but then a better performance is achieved by MLUBET for PICE values higher than 0.2. For MG and usauto datasets, M2LUBET is consistently better for a large PICE region, while MLUBET outperforms M2LUBET in the same consistent way for pass. The estimated hypervolume values are presented in Table 4, revealing that each method has three best results (cradfq, pass and water for MLUBET; MG, gas and usauto for M2LUBET) and there are three ties. However, the median over all time series favors M2LUBET by 0.06 percentage points.

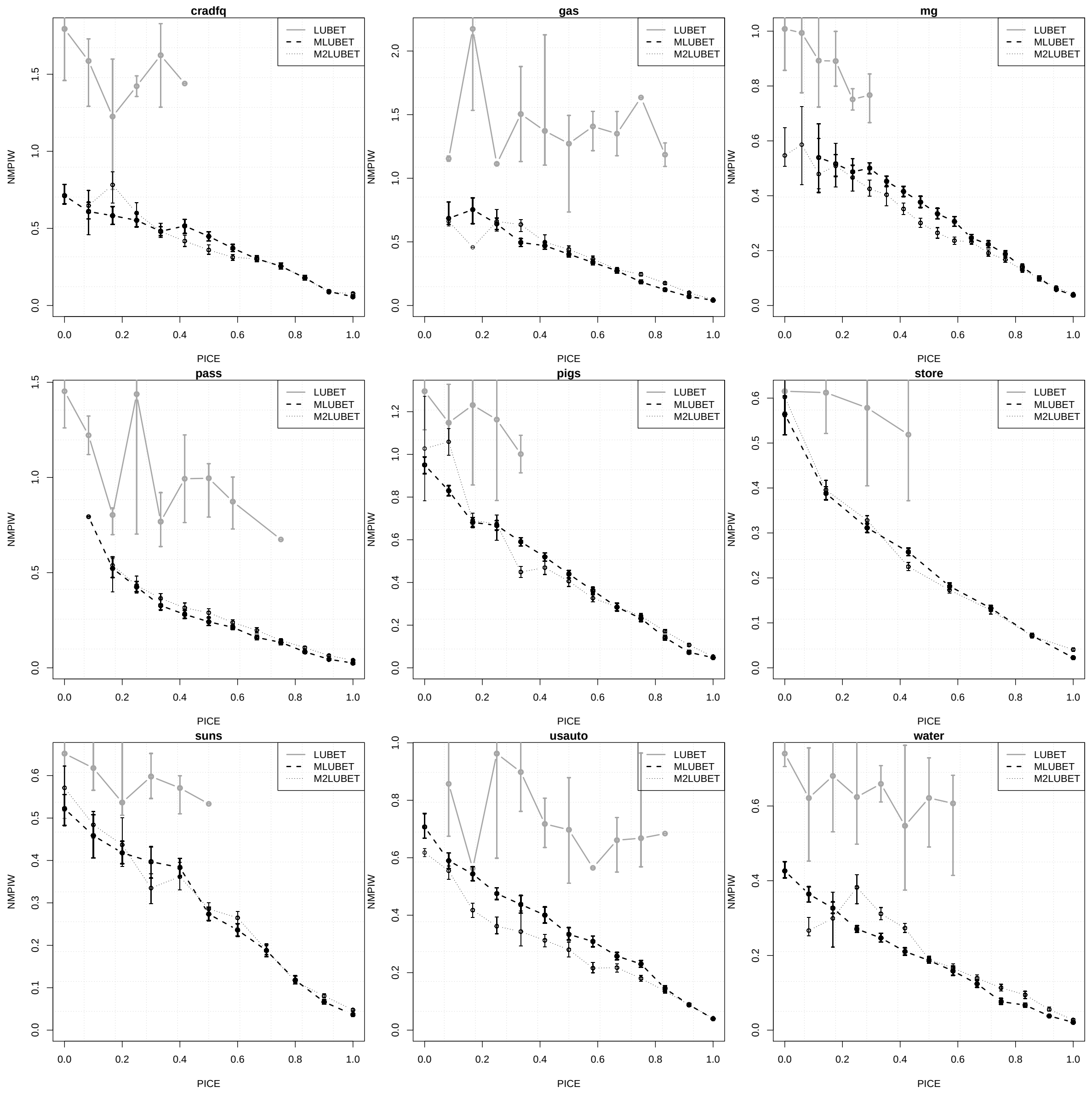


Figure 14: Forecasting results for non-ensemble methods (points denote the Wilcoxon median values and whiskers represent the respective 95% confidence intervals).

Similar results are achieved for the ensemble methods. As shown in Figure 15, the neuroevolution methods have a clear superiority when compared with LUBEXT, presenting high differences in several cases, such as cradfq, gas, pass, usauto and water. Also, the distinct neuroevolution ensembles have closer performances, whose differences depend on the time series and PICE region analyzed. For

Table 4: Median hypervolume test values for the neuroevolution methods (best values in **bold**).

	Non-ensemble		Ensemble			
Series	**MLUBET**	**M2LUBET**	**MLUBEXT**	**M2LUBEXT**	**MLUBEXT2**	**M2LUBEXT2**
cradfq	**0.58**	0.56	**0.58**	0.57	0.56	0.56
gas	0.55	**0.57**	0.56	0.47	**0.57**	0.48
MG	0.60	**0.68**	0.60	0.65	0.62	0.62
pass	0.65	0.61	0.63	0.61	**0.66**	0.59
pigs	**0.52**	**0.52**	**0.52**	0.51	**0.52**	0.48
store	**0.73**	**0.73**	0.72	0.72	0.72	0.72
suns	**0.69**	**0.69**	0.68	0.68	0.68	0.67
usauto	0.62	**0.69**	0.59	0.63	0.61	0.67
water	**0.79**	0.74	**0.79**	0.75	0.78	0.75
median	0.62	**0.68**	0.60	0.63	0.62	0.62

instance, the two-phase learning ensembles (M2LUBEXT and M2LUBEXT2) are consistently better than other neuroevolution methods for MG data, while the single phase methods (MLUBEXT and MLUBEXT2) are significantly better for the water series. In addition, M2LUBEXT2 provides the best performance for usauto data. The type of region split used for ensemble selection (radial slices versus clustering) does not seem to produce distinctive forecasting behaviors. In several cases, the respective median Pareto curves are aligned. For example, MLUBEXT and MLUBET2 show very similar curves for cradfq, MG, gas, usauto and water. The same similarity effect is visible for M2LUBEXT and M2LUBEXT2 on the series pass and water. The global hypervolume values (Table 4) confirm that there does not seem to be any single superior strategy for ensemble creation. In effect, the best hypervolume performance depends on the dataset analyzed: MLUBEXT – cradfq and water; MLUBEXT2 – gas and pass; and M2LUBEXT – MG; M2LUBEXT2 – usauto. The median values over all series (last row of Table 4) denote small differences, with the overall median values ranging from 0.60 (MLUBEXT) to 0.63 (M2LUBEXT).

When considering the comparison of non-ensemble versus ensemble methods, the results from Table 4 tend to favor the former ones. Indeed, MLUBET or M2LUBET present identical or superior median hypervolume. The only exception occurs with the pass series, where MLUBEXT2 produces the best value. Moreover, MLUBET is only outperformed by its ensemble variants in three cases (MG, gas, pass), while M2LUBET is only surpassed in two datasets (cradfq and water). In addition, the median values over all series show identical (MLUBET and MLUBEXT2) or superior (MLUBET versus MLUBEXT; M2LUBET versus any of its ensemble variants) performances. When considering all methods and hypervolume values, the best results are obtained by M2LUBET, since it ranks first in six of the nine tested datasets and it also provides the highest overall median value.

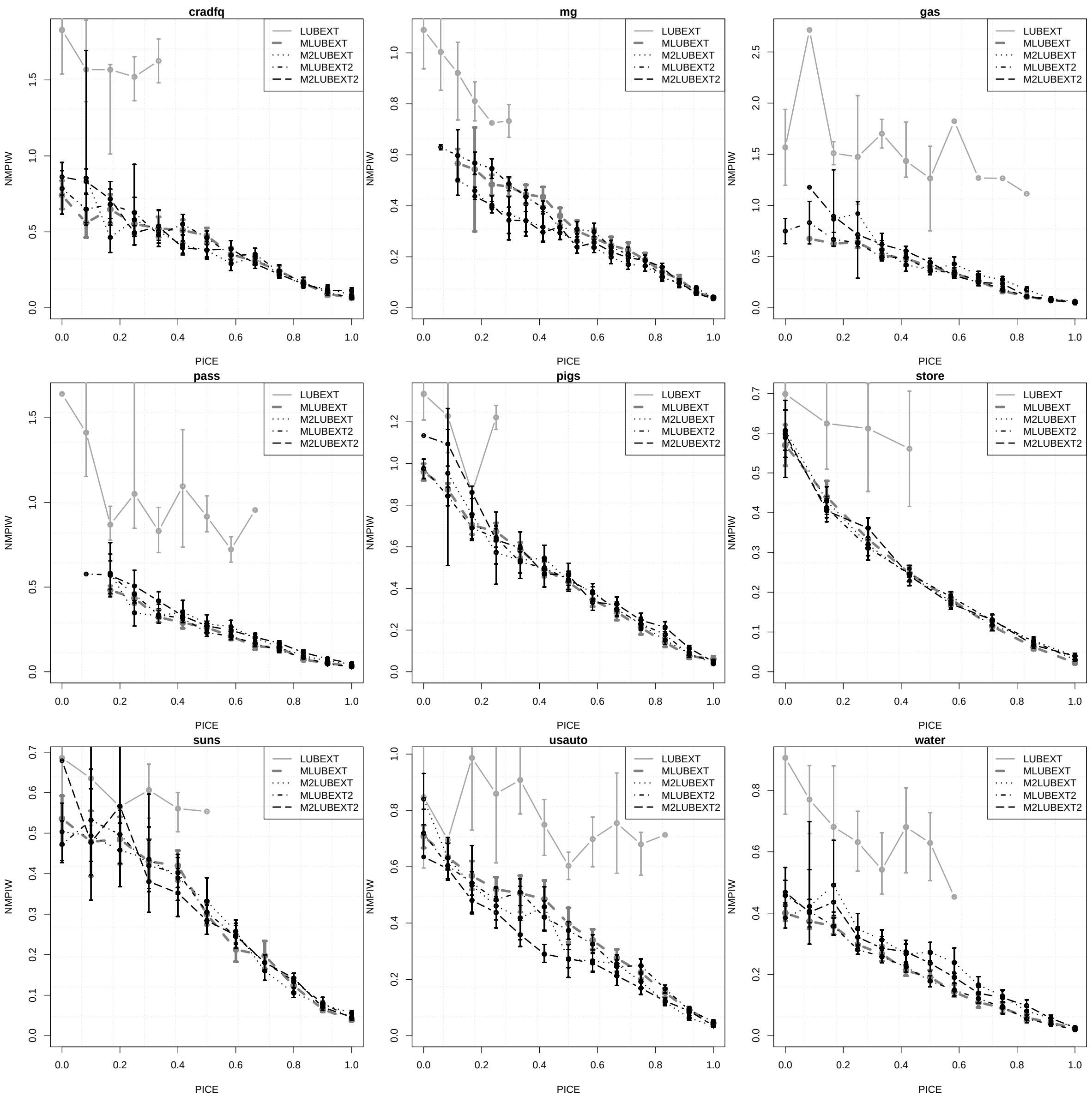


Figure 15: Forecasting results for ensemble methods (points denote the Wilcoxon median values and whiskers represent the respective 95% confidence intervals).

To demonstrate the achieved multi-step forecasts, Figure 16 shows examples of twelve multi-step ahead PI that were computed for two series. The top plots present the MLUBEXT2 predictions for the pass series, while the bottom present the M2LUBET predictions for the usauto data. For each series, two types of forecasts are exemplified (left graphs have smaller PICE values, right graphs have larger ones), allowing to visually compare different coverage-width trade-offs.

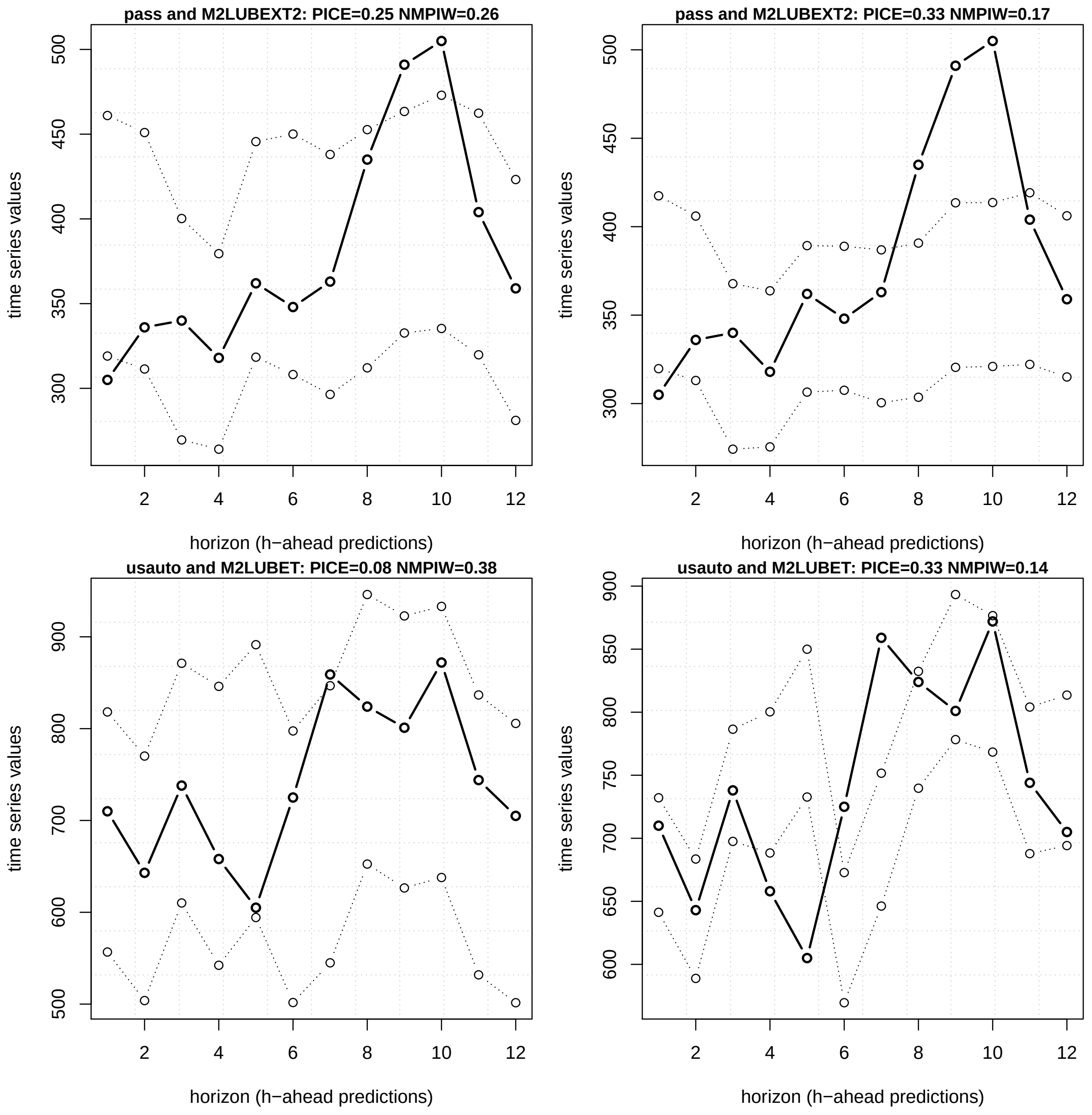


Figure 16: Examples of the obtained multi-step ahead prediction intervals (solid line denotes the true values; dashed lines represent the upper and lower prediction intervals).

3.3.3 *Computational effort and training optimization examples*

The extra computational effort for producing ensembles is negligible when compared with non-ensemble PI method training. Thus, we compare only the computational demand of each PI base learner method and whole rolling window procedure, as detailed in Table 5. The time elapsed values confirm that single-objective methods (LUBET) are faster, requiring around half of the computational effort when compared with the neuroevolution methods. This was an expected result, since LUBET base methods only use one hidden node while other methods use larger H values (Table 3). Furthermore, M2LUBET base methods require around 20% more computational power when compared with the MLUBET ones. More importantly, the computational effort for all PI methods seems highly correlated with the series length, presenting smaller values for the shorter time series (pass) and the highest value for the largest one (MG). Also, it should be noted that the required effort is reasonable for current computer processors. For instance, a single iteration of the rolling window procedure requires just around 8 minutes (498 s) for the largest series (MG) and most demanding method (M2LUBET).

Table 5: Computational effort for all the base learner methods (in seconds, rows are sorted according to increasing **Lenght** values).

Series		Base Learner Methods		
Name	**Length**	**LUBET**	**MLUBET**	**M2LUBET**
pass	144	1936	3009	3997
pigs	188	2341	4285	5130
gas	192	2379	3977	5306
cradfq	240	2810	5036	6463
store	257	3064	6060	7475
usauto	264	2804	5139	6338
water	276	2863	5376	6721
suns	289	3066	5759	7317
MG	783	5718	11686	14950
median	257	2810	5139	6463

To demonstrate the quality of the neuroevolution optimization, Figure 17 presents examples of the Pareto front convergence for series cradfq and the MLUBET and M2LUBET methods during the first rolling window iteration. In the plots, lines denote the full Pareto front while individual points represent a PI coverage-width trade-off. Also, a gradient coloring was used and that ranges from light gray (first generation) to black (last generation). Both plots reveal a substantial Pareto curve improvement over the optimization. In particular, the training hypervolume increased from 0.54 to 0.83 (MLUBET) and from 0.60 to 0.92 (M2LUBET), when considering the first and last generations of the NSGA-II evolution. Moreover, the final Pareto curves are nonlinear and mostly convex, meaning that NSGA-II managed to optimize, under a single pass, an interesting range of PI trade-offs that would outperform any linear weighted combination method (e.g., $\alpha \times PICE + (1 - \alpha) \times NMPIW, \alpha \in [0, 1]$).

3.4 Conclusions

Time series forecasting (TSF) is an important field of research that models past temporal patterns of a phenomenon (e.g., production levels or sales) in order to predict its future values. In particular, multi-step ahead forecasts, computed several time periods in advance (e.g., weeks or months), are useful to

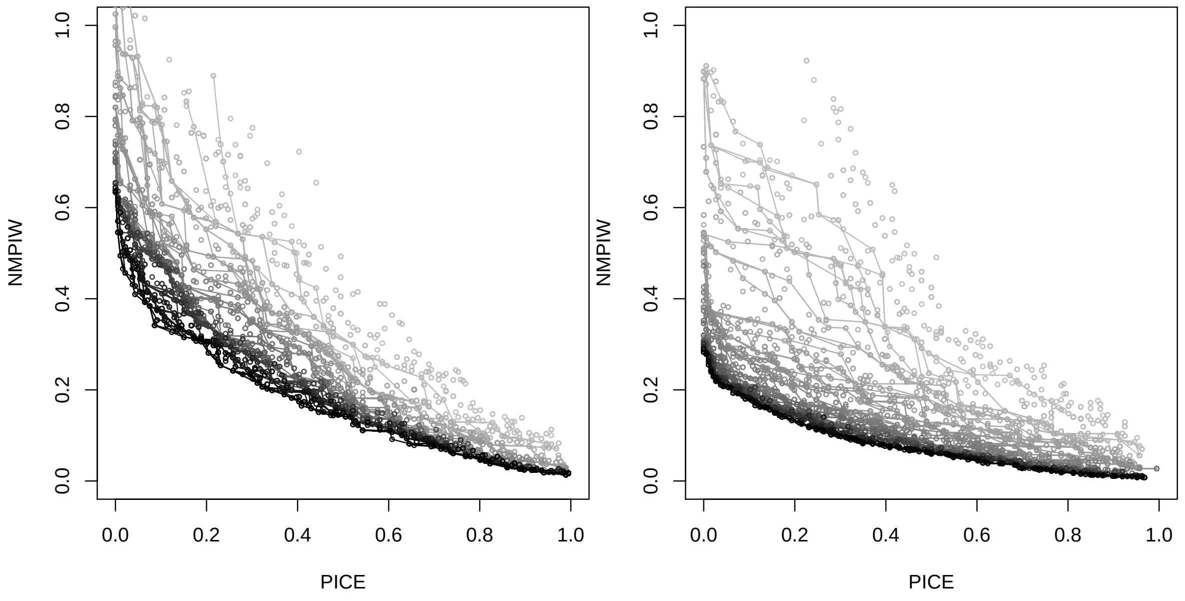


Figure 17: Evolution of the Pareto front for MLUBET (left) and M2LUBET (right) methods and cradfq series.

support tactical decisions, such as planning production resources. Given the importance of TSF, several single point prediction methods have been proposed, including statistical (e.g., Holt-Winters, ARIMA) and soft computing (e.g., neural networks, support vector machines, fuzzy techniques) approaches. However, there has been much less research that deals with TSF prediction intervals (PIs). And PIs are useful to reduce uncertainty associated with decision making. For example, it can be used to define best and worst what-if scenarios related with strategic decisions.

This chapter addresses this research gap, focusing in soft computing approaches for multi-step TSF PIs. In particular, we adapt and compare a comprehensive set of neural network methods that directly optimize PIs, which includes the lower upper bound estimation method for multi-step TSF (LUBET), the LUBET ensemble extension (LUBEXT) and two neuroevolution methods, namely the multi-objective evolutionary algorithm LUBE (MLUBET) and a two-phase learning MLUBET (M2LUBET). We also present new neuroevolution ensemble variants based on two multi-objective split methods: radial slices (MLUBEXT and M2LUBEXT) and clustering (MLUBEXT2 and M2LUBEXT2).

The PI methods were compared using a robust evaluation that considered a rolling windows procedure, nine time series with distinct characteristics and from distinct real-world domains, two PI criteria (coverage and width) and the Wilcoxon statistic. The obtained results reveal a superior performance of the neuroevolution multi-objective methods when compared with the single objective methods (LUBET and LUBEXT), even when considering the range of models associated with smaller coverage errors. In general, the non-ensemble variants produced similar or slight better results when compared with their ensemble versions. Overall, the best results were achieved by M2LUBET. This method requires a slight computational effort increase when compared with MLUBET. Nevertheless, such effort is still affordable (e.g., requiring around 8 minutes of computation to process a time series with around five hundred elements) and thus we recommend M2LUBET as the best option for multi-step TSF PI with neural networks.

In future work, we intend to research if better PI ensemble performances can be achieved by exploring other PI aggregation functions, such as usage of an average weighting or stacking (Witten et al., 2011). Also, to speedup the computation, we wish to explore multi-core parallel processing (e.g., island models) (Sudholt, 2015).

We highlight that the work described in the current chapter was developed during a preliminary stage of this doctoral project, when the business requirements from the OLAmobile company, associated with the *PRediction and Optimization of MObile Subscription marketing campaigns (PROMOS) Research and Development (R&D)* project, were being discussed and there was no Mobile Performance Marketing data available for research purposes. Although the developed TSF PI methods could be implemented for the analyzed Mobile Marketing domain (e.g., predicting the total number of redirects and sales), such approach was not considered a priority by the OLAmobile company. In effect, the scope of the PROMOS project was based on a distinct *Machine Learning (ML)* task: mobile user *Conversion Rate (CVR)* prediction. Nevertheless, this was our first work in the multi-objective optimization of predictive models that led to an important scientific contribution. Moreover, some of the researched methods were adopted in the following PhD user CVR work (presented in Chapter 4). In particular, we used the same NSGA-II multi-objective algorithm to create and optimize Decision Trees, becoming the search engine of the adopted *Grammatical Evolution (GE)* (Ryan et al., 1998). Furthermore, we used the innovative and robust evaluation approach described on this chapter, with multiple iterations rolling windows procedure, two conflicting goals and a novel vertical result aggregation using a Wilcoxon statistic test.

4

MULTI-OBJECTIVE GRAMMATICAL EVOLUTION OF DECISION TREES FOR MOBILE MARKETING USER CONVERSION PREDICTION

4.1 Introduction

The massive usage of mobile devices (e.g., smartphones, tablets) is increasing the value of the mobile advertising industry, which was estimated at 100 billion dollars worldwide in 2016 (Du et al., 2016). In the particular domain of Mobile Performance Marketing, users are matched to advertisements through *Demand-Side Platforms (DSPs)*, involving several types of mobile market players: users, publishers and advertisers. Publishers own popular digital spaces (e.g., news websites, online game services) that attract a vast audience of users to their content. Advertisers own marketing campaigns regarding products or services that they want to sell. DSPs allow the market to function by linking publishers to advertisers through a digital platform. Publishers can be funded by requiring users to click in a dynamic ad prior to accessing their contents. If a user activates a dynamic link then a redirect data event is generated. In these markets, compensation only occurs when there is a conversion, when a product or service is acquired by the user. If there is a conversion, a portion of the advertiser's profit is automatically returned to the publisher, and the DSP company also receives a base fee. Thus, the DSP goal is to perform a good match between users and advertisement campaigns, in order to increase conversions. Under this context, a key issue for the implementation of a DSP expert system is the design of a prediction model for the user *Conversion Rate (CVR)*, which is often modeled as a binary classification task ("sale", "no sale"), aiming to estimate if a user will produce a conversion once a redirect occurred.

The mobile user CVR prediction task is nontrivial due to four main reasons (Matos et al., 2019a): it involves big data, with millions of redirects being generated every day; most redirects (e.g., 99%) do not result in conversions (it is a highly unbalanced task); only a limited set of input features are available (due to privacy and technology issues); and most features are categorical and contain a high cardinality (with hundreds or thousands of levels). Despite these issues, several attempts for CVR prediction have been performed, using different types of *Machine Learning (ML)* models. The first attempts involved more rigid and linear models, such as Poisson regression and Logistic Regression (Chen et al., 2009).

These models are easy to interpret but usually provide limited prediction performances. Thus, recent CVR prediction studies use more flexible ML algorithms, such as: Random Forests (Du et al., 2016); Gradient Boosting Decision Trees (Zhang et al., 2014); Bagging, Stacking and Voting ensembles (King et al., 2015); XGBoost (Matos et al., 2018); Deep Learning (Matos et al., 2019a); and Neuroevolution (evolutionary optimization of neural network models) (Pereira et al., 2019). Yet, all these flexible ML approaches use black-box prediction models (Cortez and Embrechts, 2013), which are more difficult to be interpreted by Mobile Marketing domain experts. In effect, model interpretability, often termed *Explainable Artificial Intelligence (XAI)* (Arrieta et al., 2020), is a key element that helps to determine if a prediction model makes sense and can be trusted. Moreover, a human interpretable model can be used in posterior analysis (e.g., to help in the design of successful future marketing campaigns).

Decision Trees (DTs) are well-known ML models, particularly used in classification tasks (e.g., CART, ID3, C4.5 algorithms), due to their fast training time and good interpretability (Hastie et al., 2009; Witten et al., 2011). However, in complex classification tasks the predictive performance is often lower than achieved by other ML methods. To improve the classification results, one research direction was to propose DT ensembles, in which several trees are combined into a single model. This resulted in popular predictive models (e.g., Random Forest, XGBoost) but at the cost of losing interpretability. Another research approach is to adopt a single decision tree and improve the fitting algorithm in order to provide a higher predictive performance. For instance, by using *Evolutionary Computation (EC)*, such as proposed in (Motsinger-Reif et al., 2010; Fitzgerald et al., 2015; Rivera-López and Canul-Reich, 2018; Chabbouh et al., 2019; Czajkowski and Kretowski, 2019a).

According to Barros et al. (2012), there are two main types of evolutionary design of DTs for classification: axis-parallel and oblique. In the former, a single attribute is used to split the data in each node, while in the latter there is a combination of two or more attributes in each split node. In some studies (Czajkowski and Kretowski, 2010; Barros et al., 2011), a combination between these two is used, named mixed trees, where each split node can contain either a single attribute or a combination of multiple attributes. Axis-parallel DTs are popular in literature because they are easier to interpret when compared with oblique DTs (Barros et al., 2012). This work follows the research direction of using EC to evolve axis-parallel DTs. The associated state-of-the-art works are summarized in Table 6, ordered by publication year and with some characterizing elements:

T – the type of DT (Axis-Parallel, Oblique or Mixed);

EC – the type of EC algorithm used;

V – if a variable-length DT representation was adopted;

GOAL – the optimization main goal;

CM – the adopted DT complexity measure;

MO – if a *Multi-Objective Optimization (MO)* algorithm was used;

LE – if a *Lamarckian Evolution (LE)* was included;

DATA – the highest dataset size (total number of instances).

Table 6: Summary of the related work. The following are the keywords used in the table:
a Decision Tree type : A – Axis-Parallel, M – Mixed, O – Oblique.
b Evolutionary Computation method: DE – Differential Evolution, GDT – Global Decision Tree, GE - Grammar Evolution, GP - Genetic Programming, EA – Evolutionary Algorithm.
c Variable-length Decision Tree (DT) representation.
d Goal: PP – Predictive performance, MC – Model Complexity.
e Complexity Measure: C – coding region size, D – Tree depth, N – number of nodes.
f Multi-objective Optimization method: N2 – NSGA-II, N3 – NSGA-III.
g Lamarckian Evolution (LE).
h Highest dataset size (K means thousands of records).

Study	**T**[a]	**EC**[b]	**V**[c]	**Goal**[d]	**CM**[e]	**MO**[f]	**LE**[g]	**Data**[h]
Czajkowski and Kretowski (2010)	M	EA		PP,MC	N			22.54K
Motsinger-Reif et al. (2010)	O	GE	✓	PP				1.00K
Barros et al. (2011)	M	GP	✓	PP,MC	N			4.94K
Czajkowski and Kretowski (2013)	O	EA		PP,MC	N			8.65K
Jankowski and Jackowski (2014)	A	EA		PP,MC	D			12.83K
Fitzgerald et al. (2015)	M	GE	✓	PP				1.00K
Rivera-López and Canul-Reich (2018)	A	DE		PP				5.47K
Chabbouh et al. (2019)	O	EA		PP		N3		4.17K
Czajkowski and Kretowski (2019a)	A	GDT	✓	PP,MC				0.25K
Czajkowski and Kretowski (2019b)	O,M	EA		PP,MC	N	N2		58.72K
This work	A	GE	✓	PP,MC	C,N	N2	✓	6360.10K

This work, represented by the last row of Table 6, explores a *Grammatical Evolution (GE)* approach (Ryan et al., 1998). GE shares some similarities with *Genetic Programming (GP)* (Koza, 1993) and *Gene Expression Programming (GEP)* (Ferreira, 2001), since all these approaches optimize programs (Guogis and Misevicius, 2014). The main difference is that GE evolves programs in an arbitrary language based on a grammar. In past studies, GE has been applied to optimize ML models with different application purposes, including the creation of neural logic networks for bankruptcy prediction (Tsakonas et al., 2006) and the automatic design of Neural Networks for classification tasks (Ahmadizar et al., 2015). GE is particularly suited for variable-length solution representations, which is the case of a DT. Indeed, GE was used to evolve DTs in (Motsinger-Reif et al., 2010; Fitzgerald et al., 2015), outperforming standard DT algorithms (e.g, C4.5, CART) in several classification tasks. The advantage of using GE is that no limiting threshold needs to be set *a priori*, which is a limitation of the fixed-length tree representations used in (Rivera-López and Canul-Reich, 2018; Chabbouh et al., 2019). An important distinctive aspect of our work is the type of EC goal. Most related works from Table 6, including the ones that use GE (Motsinger-Reif et al., 2010; Fitzgerald et al., 2015), only focus on predictive performance and not interpretability. These two goals are usually conflicting and thus a trade-off often needs to be set. In (Czajkowski and Kretowski, 2010; Barros et al., 2011; Jankowski and Jackowski, 2014; Czajkowski and Kretowski, 2013, 2019a), this issue was addressed by using a single fitness function with an additive weighted formula. The problem with this approach is that it is only possible to optimize a single trade-off on each run and the fitness weights need to be set in advance.

A more natural approach, followed in our work, is to adopt a MO using a Pareto front, simultaneously maximizing the predictive performance and minimizing the DT complexity. In effect, we adopted the MO GE proposed by (Colmenar et al., 2011), that adapted the *Non-dominated Sorting Genetic Algorithm II (NSGA-II)* algorithm (Srinivas and Deb, 1994) to be the evolutionary engine of the GE. As far as we know, there are no studies that use a MO approach to optimize both classification performance and complexity for axis-parallel DTs. A MO was used in (Chabbouh et al., 2019) to evolve DTs but it only optimized predictive performance measures (Precision and Recall), while in (Czajkowski and Kretowski, 2019b) a MO was adopted to optimize oblique and mixed DT model complexity and predictive performance for regression tasks. Moreover, LE can use a local learning procedure to accelerate evolution, where the improved solution is encoded back into the chromosome (Cortez et al., 2002). Our work is the only study that introduces a LE, which uses a fast local ML search to improve the GE solutions. In Mingo et al. (2013), a similar approach was used with GE but with a non supervised local learning procedure applied in a reinforcement learning context. Finally, we note that all related work studies from Table 6 worked with datasets with a few hundred or thousands of examples. The specific mobile CVR prediction task addressed in this paper involves a higher magnitude of order, namely millions of training records. To cope with such "big data", in the sense that the datasets that are too big to be dealt with standard evolutionary DT methods, our GE approach includes two specific mechanisms: the use of a balanced sampling over the training data during the fitness evaluation; and a parallel evaluation of the population individuals by means of multi-core processors. As shown in Section 4.3, this allows to deploy a timely feasible solution for the analyzed Mobile Marketing domain.

The reason for minimizing the DT complexity is twofold. Firstly, a less complex tree will imply a better model interpretability, since it will be easier to understand by domain experts. Secondly, trees encoded by a lower number of genes are faster to build, thus reducing the prediction time used by the algorithm. This last reason is very important for the Mobile Marketing domain, since we aim to perform predictions in real-time.

In this chapter, we propose a MO based GE to evolve DTs for the Mobile Performance Marketing domain. To measure the effect of a LE, we explore two main variants: a pure GE method (MGEDT) and a GE that uses local learning to further improve the evolved solutions (MGEDTL). Both MGEDT and

MGEDTL are tested using recent real-world DSP data, provided by a marketing company and compared with traditional decision trees, a Random Forest and Deep Learning. The main contributions are:

I) we propose a MO approach that optimizes both the classification performance and model interpretability using GE under two main variants (MGEDT and MGEDTL);

II) we adopt a robust experimentation procedure, using big data from a real-world Mobile Marketing DSP provider (with 6 million data records) and a realistic rolling window evaluation (Oliveira et al., 2017), with several training and test iterations;

III) we compare the proposed GE methods with a standard decision tree, a Random Forest and a state-of-the-art Deep Learning method (Matos et al., 2019a); and

IV) we discuss how the best evolved DT is useful in the analyzed application domain (Mobile Performance Marketing).

Current chapter is organized as follows. Section 4.2 presents the Mobile Performance Marketing data, the classification algorithms (including MGEDT and MGEDTL) and the evaluation procedure. The results are presented and analyzed in Section 4.3. Finally, Section 4.4 draws the main conclusions and suggestions of future work.

4.2 Materials And Methods

4.2.1 *Mobile Performance Marketing data*

This research was conducted during a R&D project that involved a worldwide Mobile Marketing company (OLAmobile). The collected data was retrieved from the company data center cloud system, containing two main data events: redirects and sales. A redirect event record is generated each time a user clicks on a dynamic link related with an advertisement. The user is forwarded to a mobile marketing campaign. A sale event only occurs if the redirect originates a conversion (e.g., product purchase or service subscription). The DSP supported by the marketing company generates millions of redirects per day. Only a few thousands of these redirects produce a sale.

The DSP from OLAmobile works under two different traffic models: BEST and TEST. In an initial phase, new marketing campaigns are put in a TEST mode, in which the DSP randomly assigns an ad to a user. Then, after obtaining a minimum number of conversions, the best campaigns are put in a BEST mode, in which the DSP uses simple statistics computed using past results to match ads with users. For instance, it can select the ad with the best previous sales for a particular mobile carrier. We collected all the DSP data events that were generated during a four day period, from 15th to 18th of November of 2019 (Table 7). Most of the analyzed DSP traffic belongs to the BEST mode (64.5%).

Only a tiny fraction of the user clicks result in a conversion. As expected, a higher CVR is produced with BEST traffic (2.8%) when compared with the TEST mode (1.0%). User CVR prediction is thus a highly unbalanced task.

Table 7: Summary of the collected Mobile Marketing data.

Mode	**Redirect events**		**Sale events**	
BEST (64.5%)	4,076,375	(97.2%)	112,532	(2.8%)
TEST (35.5%)	2,283,725	(99.0%)	22,769	(1.0%)
Total	6,360,100	(97.9%)	135,301	(2.1%)

After collecting data, we merged the two data event records for each traffic mode. As explained in the previous section, due to privacy issues and technological constrains, the number of collected Mobile Marketing features is quite reduced. For instance, it is not possible to identify a single user and memorize any of its temporal purchasing behaviour. Thus, the DSP can only work with indirect and immediate context features (retrieved once the user clicks a dynamic link), which are presented in Table 8. There are a total of ten input attributes related with the user, publisher and advertiser (column **Context**) and one target binary output (Y). All inputs are categorical. Some present a high cardinality, such as the mobile operator (with 446 different levels for TEST traffic and 418 for BEST) and advertiser campaign code (with 1,268 and 1,147 levels).

Table 8: Description of Mobile Marketing data attributes

Context	**Attribute**	**Description** (TEST levels, BEST levels, examples)
user	country	user country (222, 219, {`"Brazil","Spain"`})
	region	user region (22, 24, {`"Asia","Europe"`})
	browser	browser name (13, 14, {`"Chrome","Safari"`})
	operator	mobile carrier (446, 418, {`"Vodafone","WiFi"`})
publisher	partner	publisher code (160, 162, *numeric*)
	account	publisher type (10, 9, {`"Marketer","Network"`})
	manager	manager code (10, 11, *numeric*)
advertiser	campaign	ad campaign code (1268, 1147, *numeric*)
	vertical	ad type: 11 and 12 (e.g., {`"video","mainstream"`})
	application	product code (729, 785, *numeric*)
target	Y	if there is a sale (2, 2, {`"yes","no"`})

A high categorical cardinally increases the computational complexity, in terms of both memory and processing effort of the ML algorithm. Thus, during the preprocessing step, we applied the *Inverse Document Frequency (IDF)* transform to all inputs. The IDF transform was proposed in (Campos et al., 2016) and produced good results in our previous work (Pereira et al., 2019). This transform converts a categorical feature into a numeric value by assigning each level according to $IDF(x) = \ln(\frac{N}{f_x})$, where N is the total number of instances and f_x is the number of occurrences of level x. Levels with higher frequency have transformed values close to 0, while levels that rarely appear tend to the maximum value of $ln(N)$, for $f_x = 1$. When applying the IDF transform, each level and respective transformation is stored in memory, thus enabling each mapping to be applied to test (unseen) data. It should be noted that due to the steady flow of new campaigns, unseen levels can appear on the test data. These new levels were replaced by the highest $ln(N)$ value found in the training data, which corresponds to an infrequent training set IDF value. Our IDF implementation is available at the cane Python module[1].

4.2.2 *Multi-objective Grammatical Evolution Decision Trees*

GE is an evolutionary algorithm proposed in 2001 by O'Neill and Ryan (O'Neill and Ryan, 2001). It is capable of evolving a program that is based on a provided grammar. A GE starts by generating an initial population of solutions, often randomly generated, each corresponding to an array of integers (genome), that will be used to generate the program (phenotype). GE uses an evolutionary algorithm, which includes two phases in each generation (iteration): evolution and evaluation. The latter consists in evaluating the population of solutions, based on a provided fitness function, while the former is composed by a set of operations: selection, crossover and mutation. In a GE algorithm, evolution is applied directly to the genome, while evaluation is applied to the phenotype. Thus, to obtain the phenotype from the evolved genome, GE applies a mapping process between the two phases.

The GE mapping process uses the genome integer values to select rules from a context-free *Backus–Naur Form (BNF)* grammar, which contains a set of symbols, that can be terminal or non-terminal, each of which is composed by a set of production rules. Terminal symbols are items that can appear in the language (e.g, +, -), while non-terminals are variables that can be expanded in one or more terminal or non-terminal symbols (O'Neill and Ryan, 2003). The mapping process begins at the start symbol and stops when a complete program is formed, i.e., there are no more non-terminals to be expanded. The procedure implies expanding each non-terminal by selecting the production indicated by the chromosome. For each value V and if there are P productions (numbered $0, 1, \dots, P-1$) for a given non-terminal, the R-th production is used where R is the remainder of the division of V by P.

In this work, we developed *Multi-objective Grammatical Evolution Decision Tree (MGEDT)*, a multi-objective algorithm that uses GE to create and evolve DTs for Mobile Marketing user CVR prediction. A DT is composed by a set of decision nodes, where binary (true or false) conditions are applied, defining a path until a leaf node is reached, which returns the expected output. We feed our *Evolutionary Decision Tree (EDT)* with a set of data records. For each record and decision node, we compare a value

1 https://pypi.org/project/cane/

from a specific feature with a constant value, defined by GE, until a leaf is achieved. A leaf returns the probability p of occurring a conversion, i.e., the probability of class “sale”, with $p \in [0, 1]$. Our BNF grammar encodes a Python subset that is capable of representing a DT that returns the sale probability p (for $Y =$ yes). The rationale behind showing our grammar and the subsequent Python implementation is to illustrate the power of GE. When compared with GP or GEP, GE has the advantage of allowing an easy customization to a given problem or programming language by simply modifying the grammar used. Thus, the associated GE phenotype is a piece of Python code that can be executed in order to obtain the predicted sale probability p.

⟨*result*⟩ ::= numpy.where(x[⟨*idx*⟩] ⟨*comparison*⟩ ⟨*value*⟩, ⟨*result*⟩, ⟨*result*⟩)
 | (⟨*leaf*⟩)

⟨*value*⟩ ::= ⟨*digits*⟩.⟨*digits*⟩ | ⟨*digits*⟩

⟨*leaf*⟩ ::= 0.⟨*digits*⟩ | 1

⟨*digits*⟩ ::= ⟨*digits*⟩⟨*digit*⟩ | ⟨*digit*⟩

⟨*digit*⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

⟨*comparison*⟩ ::= == | < | > | <=

⟨*idx*⟩ ::= `country` | `region` | ... | `application`

The `numpy.where(`*cond*, *expression1*, *expression2* `)` is a Python function that computes a typical if-else control instruction, returning *expression1* if *cond* is true, else it returns *expression2*. The comparison used in *cond* assumes the typical Python logical operators (e.g., ==, <). The object x represents a vector with the IDF values of the redirect input attributes from Table 8 (<*idx*>) for a particular instance or redirect.

For example, the expression `numpy.where(x[country]<1,0,1)` returns 0 if the IDF transform of the user country is lower than 1, otherwise it returns 1. Given that the IDF encoded levels are previously stored, the decision tree node tests can be easily interpreted as membership set operations. For the same example, if the term *country* denotes the user `x[country]` value and the IDF levels lower than 1 correspond to the countries of Brazil and Spain, then the decision test sale probability rule is equivalent to: if *country* ∈ {”Brazil”,”Spain”} then 0%. The BNF grammar assumes a recursive definition <*result*>, which allows the creation of an arbitrary chain of *if else* executions, thus creating a DT. The leaves (<*leaf*>) of the tree include a real value, between 0.0 and 1.0 (p). Only zero or positive real values are allowed, which is consistent with the IDF transform used to preprocess the attributes (Section 4.2.1). The real values use a recursive definition of <*digits*>, thus it can include a variable size of digits. Section 4.3 provides an example of a tree generated by GE (Figure 20).

A brief illustration of the genotype to phenotype mapping is given below:

88	208	89	47	111	131	135	84	181	130	175	231	...

The associated expansion using the grammar is:

```
<result>
numpy.where(x[<idx>]<comparison><value>,<result>,<result>)      [1st prod., 88%2=0]
numpy.where(x["country"]<comparison><value>,<result>,<result>)  [idx 10th prod., 208%11=10]
numpy.where(x["country"]<<value>,<result>,<result>)             [comparison 2nd prod., 89%4=1]
numpy.where(x["country"]<<digits>,<result>,<result>)            [value 2nd prod., 47%2=1]
numpy.where(x["country"]<<digit>,<result>,<result>)             [digits 2nd prod., 111%2=1]
numpy.where(x["country"]<1,<result>,<result>)                   [digit 2nd prod., 131%10=1]
numpy.where(x["country"]<1,(<leaf>),<result>)                   [result 2nd prod., 135%2=1]
numpy.where(x["country"]<1,(0.<digits>),<result>)               [leaf 1st prod., 84%2=0]
numpy.where(x["country"]<1,(0.<digit>),<result>)                [digits 2nd prod., 181%2=1]
numpy.where(x["country"]<1,(0.0),<result>)                      [digit 1st prod., 130%10=0]
numpy.where(x["country"]<1,(0.0),(<leaf>))                      [result 2nd prod., 175%2=1]
numpy.where(x["country"]<1,(0.0),(1))                           [leaf 2nd prod., 231%2=1]
```

The expansion stops after the first 12 bytes, since there are no more non-terminals to expand. This corresponds to the size of the coding region. The remaining genotype corresponds to a non-coding region that is ignored.

In this work, we also propose a hybrid learning *Multi-objective Grammatical Evolution Decision Tree Lamarckian evolved (MGEDTL)* approach, which combines both global (provided by the GE) and local search (provided by a standard DT algorithm fit), aiming to improve the quality of the solutions. Our Lamarckian approach is applied to all individuals of the GE population. First, a random decision node from an individual tree T is selected and the associated subtree S is extracted. The training records that fall into subtree S are used to fit a new subtree S' by using a conventional DT algorithm. In particular, we used CART, as implemented in the popular sklearn Python ML module (using its default parameters) (Pedregosa et al., 2011). Then, the newly obtained subtree S' is inserted into a tree T', which is identical to T except that S is replaced by S'. The new obtained solution is evaluated and, if a higher predictive performance is achieved, it will replace the original individual representation. To achieve this, we have implemented an inverse mapping process that is able to obtain the genome from the phenotype, so that both can be stored and, thus, solutions generated by our local search procedure can be incorporated back into the GE. Figure 18 exemplifies how the proposed LE works.

The two GE variants (MGEDT and MGEDTL) adopt a MO approach, based on the NSGA-II adaptation proposed by (Colmenar et al., 2011). The adaptation performs a Pareto front MO and it was set to simultaneously optimize two goals (the fitness functions): the DT predictive performance, measured using *Area Under the Curve (AUC)* of the well-known *Receiver Operating Characteristic (ROC)* curve (Fawcett, 2006; Du et al., 2016; Matos et al., 2019a); and the DT complexity, measured as the number of genes used by the GE representation, i.e., the size of the coding region.

By assuming class probabilities ($p \in [0, 1]$) at the leaves (and not pure class labels), the optimized DTs resemble regression trees applied to a binary classification context. The advantage is that the obtained DTs are more flexible, allowing to easily compute the AUC measure and also define different

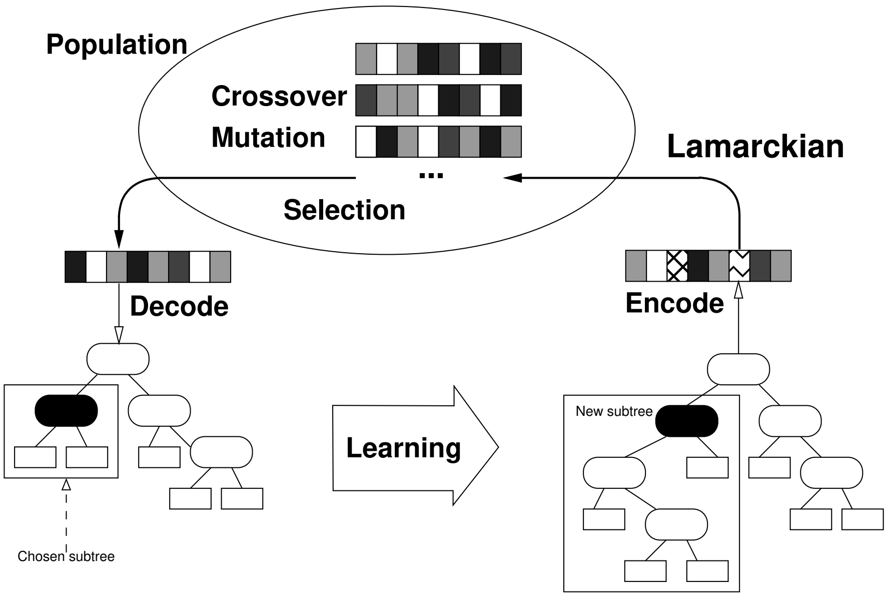


Figure 18: Schematic of the proposed MGEDTL approach.

sensitivity and specificity trade-offs for the same fitted model. Indeed, the ROC curve is a popular analysis for classification tasks (Fawcett, 2006). If a classifier outputs a class probability p then the class can be interpreted as positive if $p > K$, where K is a fixed decision threshold, otherwise it is considered negative. The ROC curve shows the performance of a binary classifier across all $K \in [0, 1]$ values, plotting one minus the specificity (x-axis) versus the sensitivity (y-axis). The overall discriminatory performance is given by $AUC = \int_0^1 ROCdK$. The AUC measure provides two main advantages: First, quality values are not affected if the classification data is unbalanced (Fawcett, 2006). Our Mobile Marketing data is highly unbalanced, thus a useless classifier that always predicts "no sale" ($Y =$ no) can obtain a classification accuracy of 99%. In contrast, the AUC value for the same "dumb" classifier would be 50%, which corresponds to a random discrimination. Second, the AUC values have a easy human interpretation. Often, the AUC values can be interpreted as (Fan et al., 2006; Gonçalves et al., 2020): 50% – performance of a random classifier; 60% – reasonable; 70% – good; 80% – very good; 90% – excellent; and 100% – perfect. The AUC is computed using several data records, thus its computation requires some effort (this issue is further addressed in the last paragraph of this section). The adopted MO implemented using the PonyGE2 Python module (Fenton et al., 2017) only handles minimization. Thus, the predictive performance goal was adapted as to minimize the symmetrical of the AUC: $-1 \times AUC$.

Regarding the complexity, we adopt the coding region size, i.e., the number of genes of the GE representation that was used for creating the tree, as a proxy of the DT complexity. Similarly to the information entropy concept proposed by Shannon (1948), we consider the coding region size an interesting complexity measure since it represents the minimum amount of integers that are needed to represent a DT. Smaller values correspond to simpler models that are therefore more easy to interpret.

There are two main advantages for using this complexity measure: its computation is very fast when using the adopted GE implementation (PonyGE2); and it puts a pressure to reduce several of the DT elements, including the number of DT nodes and also the number of digits used by the numeric constants (e.g., favoring 0.2 when compared with 0.19987). This pressure prevents bloat, which corresponds to an unnecessary growth of the program size without any improvement in the classification performance (Doerr et al., 2017).

In terms of implementation, all code was developed in the Python programming language. We used scikit-learn (Pedregosa et al., 2011) for standard DT and to calculate performance metrics, the Ply module (Beazley, 2001) for the inverse-mapping process (from phenotype to genome) and the PonyGE2 package (Fenton et al., 2017) to implement the GE algorithm. Any EC algorithm typically contains a number of hyperparameters that needs to be set by the user. In this study, we have adopted the default PonyGE2 values, which includes: subtree crossover (matching non-terminal nodes of ancestor subtrees) with a crossover probability of 75%; subtree mutation (random replacement of a subtree) with a mutation probability of 100%; selection uses the NSGA-II selection that performs a Pareto tournament selection on 2 individuals and an elitism of 10 individuals. Using previous Mobile Marketing data (collected in the year of 2018), we have performed preliminary experiments to adjust the population size to 100 and the number of generations to 100. Higher values did not improve the results but substantially enlarged the computational effort. Additionally, we incorporated the same training data sampling scheme proposed in (Pereira et al., 2019). In each generation, we randomly sample $n =$1,000 positive ($Y =$ yes) and $n =$1,000 negative ($Y =$ no) records from the training data, thus individuals are only evaluated using these 2,000 examples. This allows to deal with the unbalanced target issue and also to speed up the DT fitting for big data. Regarding the computational effort, since the evaluation of each individual is an independent process, we also employ parallelism by evaluating multiple individuals simultaneously in different cores, aiming to reduce the overall execution time.

4.2.3 *Comparison methods*

For comparison purposes, we have selected three methods: a standard DT, a *Random Forest (RF)* and a *Deep Learning (DL)* model. The DT is a white-box model based on the CART algorithm and the same IDF categorical transform, implemented in the scikit-learn Python module (Pedregosa et al., 2011). The RF is an ensemble method that combines a large set of T decision trees, where each tree depends on a random sample of features and training examples (Breiman, 2001). Similarly to the standard DT, the RF model uses the IDF categorical transform and the scikit-learn module Python implementation, set with its the default values (e.g., ensemble with $T = 100$ trees). As for the DL, it uses the state-of-the-art model proposed in (Matos et al., 2019a). This DL is based on a deep multilayer perceptron architecture with: 9 hidden layers, with 1024, 512, 256, 128, 64, 32, 16, 8 and 2 hidden nodes that use the ReLU activation function; 1 output node with the logistic function; and dropout values of 0.5 and 0.2 applied in the fourth and sixth hidden layers. The DL is trained using the AdaDelta gradient function and the stochastic gradient decent method. In terms of the categorical transform, the best DL results were achieved with a variant of the one-hot encoding called *Percentage Categorical Pruning (PCP)*, which merges the 10% less frequent levels into a single level before applying the one-hot transform. Moreover, the DL proposed includes a reuse mode that dynamically adapts the network to new batches of training data (without resetting the previous learned weights). In (Matos et al., 2019a), the DL reuse architecture outperformed both a logistic regression and a Convolutional Neural Network (CNN).

Regarding the model complexity in the DT experiments, when computing the coding of region size measure, we converted the fitted trees to our GE grammar, using the inverse mapping process that was implemented for the MGEDTL algorithm. This allows the computation of the same complexity measure (**C**) used by the MGEDT and MGEDTL algorithms, which is the number of genes in the GE coding region. Regarding the RF model, the overall coding of region size (**C**) and number of nodes (**N**) complexity measures are computed as the sum of the complexities (**C** or **N**) of the RF individual trees. Since this procedure can not be applied to DL, we opted to compute, as a reasonable alternative, the DL complexity in terms of the number of connection weights, which correspond to the total number of neural network adjustable parameters (Bianchini and Scarselli, 2014).

4.2.4 *Evaluation*

In the Mobile Performance Marketing domain, since compensation only occurs when a product or service is purchased, it is crucial to present the right advertisement to the users. This is often translated into a binary CVR prediction goal that estimates the conversion probability for a given user and is used to rank a set of active candidate advertisement campaigns. To be useful for the DSP, the prediction model should issue real-time predictions, allowing to select an advertisement in less than 10 milliseconds. Another goal is to provide a human understandable model, which will help increase the acceptance of the ML model by the DSP managers and also helps the design of future campaigns. Simpler models, which are more easy to be interpreted, also provide faster prediction responses. In this work, the prediction goal is measured by the $-1 \times AUC$ statistic and the interpretability goal is assessed by the number of GE genes in the GE coding region (denoting this as **C** measure). Both goals are minimized by the GE. Since the complexity goal can vary by different orders of magnitude, from a few dozens to hundreds of thousands, we apply the base 10 logarithm to facilitate the analysis of the complexity results. To further compare the complexity of the tree-based models, we also adopt the standard number of tree nodes (**N**) measure (Czajkowski and Kretowski, 2010; Barros et al., 2011). Given that we work with big data and there are real-time prediction requirements, we also track the training and test computational effort, measured in terms of elapsed time (in seconds).

To evaluate our models, we adopt the realistic *Rolling Window (RW)* scheme (Tashman, 2000; Oliveira et al., 2017). The RW is realistic because it simulates a dynamic scenario in which several models are trained and tested through time. It also provides several executions, allowing the computation of aggregate results using a statistical confidence measure. Similarly to what is proposed in (Cortez et al., 2020), in this paper we use the Wilcoxon nonparametric test to compute the median and corresponding confidence intervals at the 95% level. After a consultation with the Mobile Marketing experts (from the DSP company), we have set a realistic RW scheme that assumes a temporal window of 2 days of training data and a subsequent window of 6 hours of testing data. In the first iteration, the oldest 2 day redirects are used to fit the ML algorithm and then tested in the next 6 hours of test (unseen) data. In the second iteration, the training data is slided, where the oldest 6 hours of redirects are discarded and the previous test data is added. The ML is retrained and then a new set of predictions (up to 6 subsequent hours) is produced, and so on. In total, this leads to 7 iterations (column **Iter.**), as may be seen in Table 9. The RW was executed separately for each traffic type (TEST and BEST), since each traffic mode uses a different DSP matching algorithm and set of marketing campaigns.

Table 9: Description of RW data and GE generations used.

Mode	**Iter.**	**Date**	**Records**		**Generations**
			Train	**Test**	
BEST	1	17/11/19 06:00	1,069,684	163,877	100
	2	17/11/19 12:00	1,227,072	174,192	25
	3	17/11/19 18:00	1,392,913	208,410	25
	4	18/11/19 00:00	1,522,014	179,684	25
	5	18/11/19 06:00	1,494,620	167,019	25
	6	18/11/19 12:00	1,487,351	191,811	25
	7	18/11/19 18:00	1,485,581	1,040,142	25
TEST	1	17/11/19 06:00	588,413	102,745	100
	2	17/11/19 12:00	689,687	99,656	25
	3	17/11/19 18:00	787,714	121,299	25
	4	18/11/19 00:00	874,832	113,289	25
	5	18/11/19 06:00	873,808	90,969	25
	6	18/11/19 12:00	869,030	103,266	25
	7	18/11/19 18:00	865,884	577,216	25

In each of the 7 RW iterations, the IDF transform is applied to the data, taking no previous assumptions. Similarly to what is done using the reuse DL model, the GE models are readjusted between two consecutive RW iterations. This is achieved by using a seeding initialization, in which the last optimized population, obtained in the previous RW iteration, is used as the initial population of the current RW iteration. Thus, GE only needs to create completely random populations in the first RW iteration. By reusing the solutions, the GE does not need a large number of generations to achieve a quality performance (as shown in Section 4.3). Therefore, we decreased the stopping criterion, lowering the number of maximum generations to 25 after the first RW iteration (as shown in Table 9).

4.3 Results

All experiments were conducted using code written in the Python programming language. The GE and DT were executed using a dedicated Linux Intel Xeon 1.7 GHZ server, where 25 cores were used by each GE experiment. The DL experiments were conducted on a personal computer with a NVIDIA Geforce GTX 1060 GPU using the Keras and Tensorflow libraries, aiming to decrease the computational cost.

Figure 19 presents the overall RW test data results in terms of the BEST (left) and TEST (right) traffic modes. Each plot was build by using the same evaluation procedure proposed by Cortez et al. (2020). First, the test set predictions for each iteration were used to compute individual curves, in terms of the predictive performance (−AUC) and ML complexity. Then, curves were aggregated vertically, by fixing a −AUC value and computing the respective Wilcoxon median and 95% confidence intervals for the complexity. In Figure 19, the top plots present the **C** complexity (coding region size), while the bottom graphs show the **N** complexity (number of nodes, shown for all tree-based methods). To facilitate the visualization, the complexity values were transformed using a logarithm of base 10.

Figure 19 confirms that there is a trade-off between predictive performance and ML model complexity. For both MGEDT and MGEDTL, the simplest models (with less than 10 genes, **C** measure) provide a low quality performance, typical of a random classifier, while high quality predictions (e.g., AUC higher than 90%) are obtained by more complex DT, with around 10,000 genes used by MGEDTL. The two GE methods have different performances. For both traffic modes and for the same predictive performance level, MGEDT optimizes simpler models when compared with MGEDTL (with a statistical significant difference). However, MGEDT is not capable of achieving a wider AUC range such as MGEDTL does. Table 10 shows the highest AUC median values for each ML method, corresponding to the extreme left $x-$axis point values from Figure 19. From the table, it is clear that MGEDTL reaches the best AUC for BEST (92.0%, tied with RF) and second best AUC for TEST data (92.0%). In contrast, MGEDT provides the worst discrimination (when compared with other methods), reaching 80.0% (BEST traffic) and 76.0% (TEST traffic). Moreover, Figure 19 shows an interesting correlation between the **C** and **N** complexity measures, given that the tree-based models show **N** complexity differences that are similar to the ones obtained using the **C** measure. It is also interesting to notice that very competitive results were achieved by the MGEDTL approach when compared with the DT, RF and DL methods. The DL provided the third best (AUC of 89.9% for BEST traffic) and best (AUC of 93.1% for TEST traffic) predictive results. As for the RF, it obtained the best AUC result (92.0%, tied with MGEDTL) for BEST data and third best AUC value (88.2%) for TEST data. Yet, the DL and RF models are more complex (around 2/3 orders of magnitude) than the trees evolved by MGEDTL. As for DT, it provided the fourth

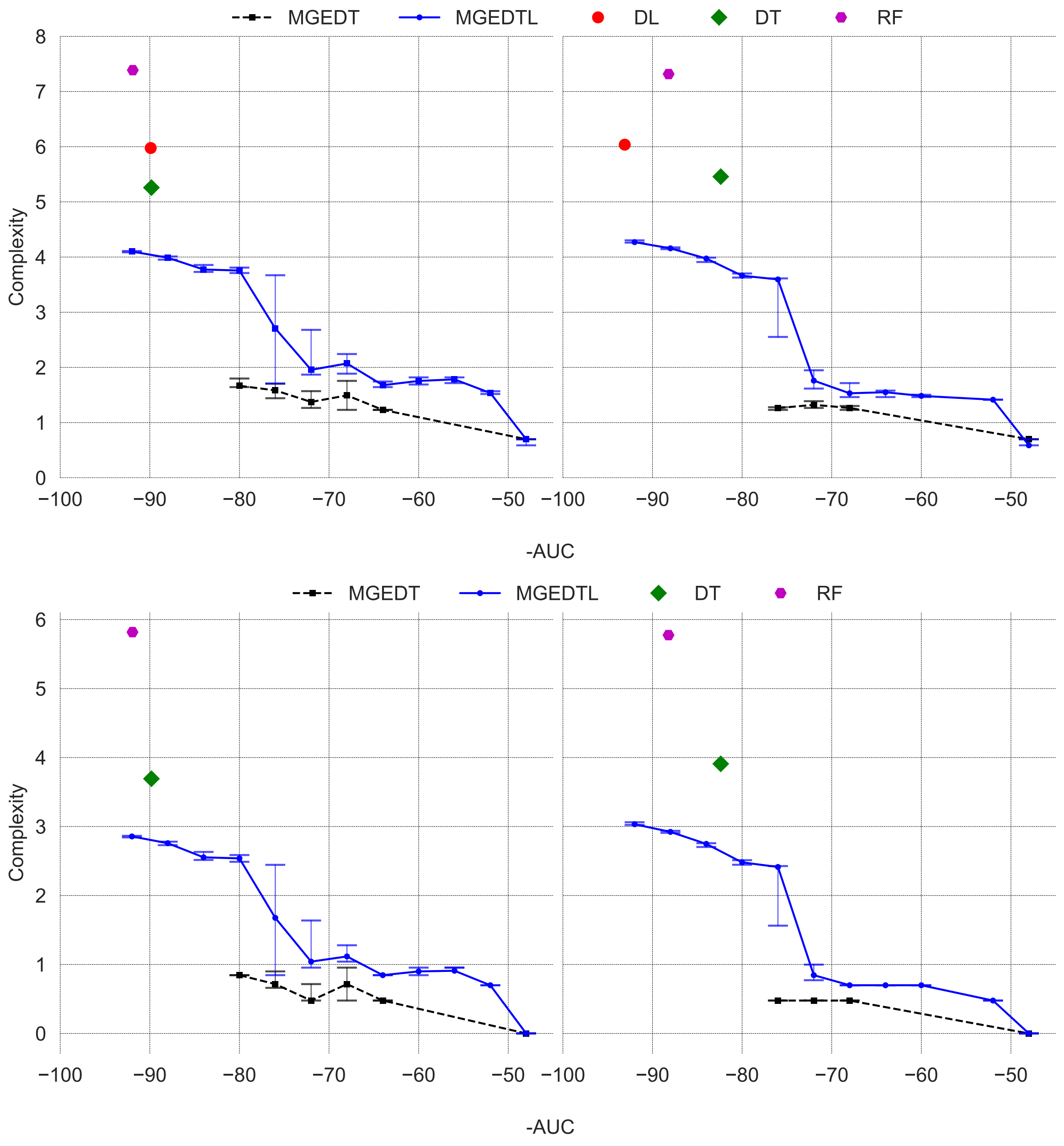


Figure 19: Complexity Wilcoxon confidence intervals of the AUC median values for all RW iterations for the BEST (left plots) and TEST (right plots) traffic modes. The top two graphs show the **C** complexity, while the bottom two graphs plot the **N** complexity (both shown at the y−axis and using a base 10 logarithm scale).

best AUC result for both traffic types (AUC values of 89.8% for BEST and 82.4% for TEST). And the fitted trees are much more complex (around 1.5 orders of magnitude) than the MGEDTL evolved ones.

Table 10: Extreme left $x-$axis AUC values from Figure 19 (best value in **bold**; relative rank of the ML method is shown in parentheses).

	ML method				
Mode	DT	RF	DL	MGEDT	MGEDTL
BEST	89.8% (4)	**92.0**% (1)	89.9% (3)	80.0% (5)	**92.0**% (1)
TEST	82.4% (4)	88.2% (3)	**93.1**% (1)	76.0% (5)	92.0% (2)

To better understand the complexity values, we selected the solutions with the highest complexity in each iteration for each tree-based model and estimated the median number of decision split nodes (i.e., `numpy.where` instructions) for the BEST and TEST traffic modes. MGEDT presents the lower median number of decision nodes (with 3 for BEST and 1 for TEST), followed by MGEDTL (with 367 for BEST and 864 for TEST) and then DT (with 10,303 for BEST and 8,413 for TEST). For demonstrative purposes, Figure 20 shows an example of a GE generated solution in terms of its Python code (left) and a plot of the associated DT (right). This solution includes a total of 4 `numpy.where` instructions, which correspond to 4 decision tree nodes. An example of an easy interpretable rule that can be extracted from the DT is: if *partner* ∈ { `"0"`} and *manager* ∈ { `"1"`} and *operator* ∈ { `"Vodafone"`, `"Orange"`, ...} then $p = 30\%$. In this rule, the `"0"` and `"1"` terms denote numeric identification labels and the set { `"Vodafone"`, `"Orange"`, ...} includes a total of 272 different mobile operators. This DT was optimized by the MGEDTL method, during the last iteration of the RW procedure (described in Section 4.2.4) for BEST traffic. The coding region has a size of 74 genes and obtained an AUC of 77%.

```
numpy.where(x['partner'] > 2.32,
    numpy.where(x['app'] < 34, (0.95), (0.6)),
    numpy.where(x['manager'] <= 1,
        numpy.where(x['operator'] <= 2, (0.5), (0.3)),
        (1)
)
```

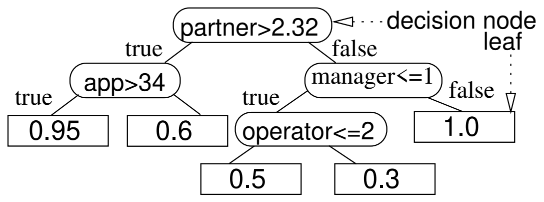


Figure 20: Example of a generated Python code (left) and the corresponding DT (right).

Two figures are presented to show the evolution of the ML algorithms through time. First, Figure 21 plots the highest AUC test values obtained for each algorithm during the RW iterations for the BEST (left) and TEST (right) data. The GE algorithms tend to have a stable performance through time. As for the DL model, it suffers a decay in performance during the third and fourth RW iterations for the BEST traffic. Aligned with overall results of Figure 19, the best global AUC performances are provided by the MGEDTL, RF and DL algorithms, with DT presenting similar best AUC results for the BEST traffic. Second, Figure 22 plots the hypervolume of the GE algorithms through the RW iterations. Hypervolume is a popular approach to compare Pareto fronts using a single measure and it is computed by considering a reference point, which is regarded as the worst possible solution (Beume et al., 2009). In this work, the reference point was set as -50% AUC and a complexity of 5 (when using the logarithm of base 10). This assumes that the perfect solution is the -100% AUC and 0 complexity point. For BEST mode (left of Figure 22), both curves are very close, with MGEDTL presenting 4 (of 7) best hypervolume values. Regarding the TEST mode (right of Figure 22), the Lamarckian variant obtains a better overall hypervolume curve.

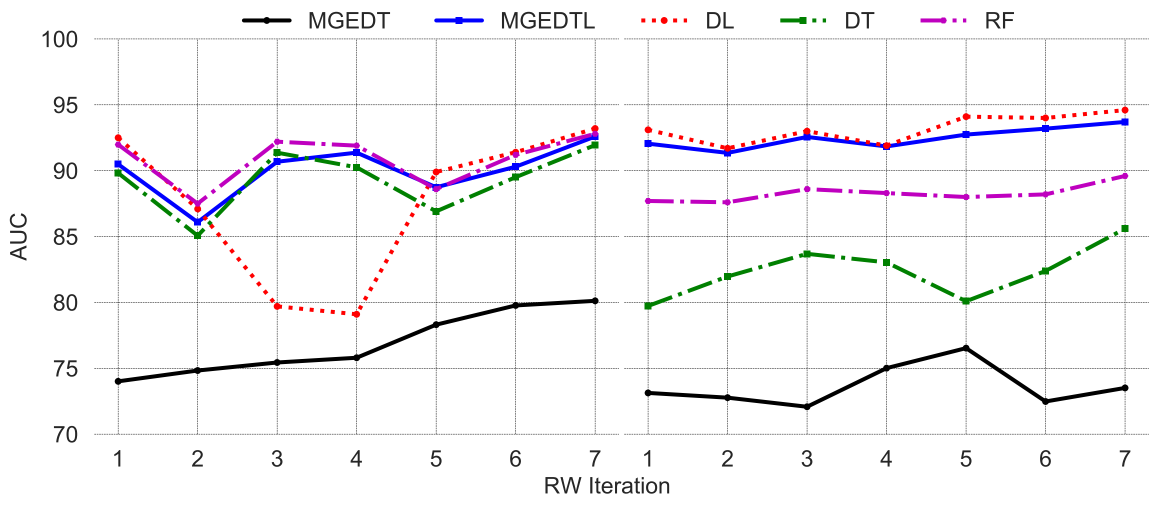


Figure 21: Evolution of the best AUC values of the ML classifiers models for BEST (left) and TEST (right) traffic (x-axis denotes the RW iteration and y-axis the AUC value).

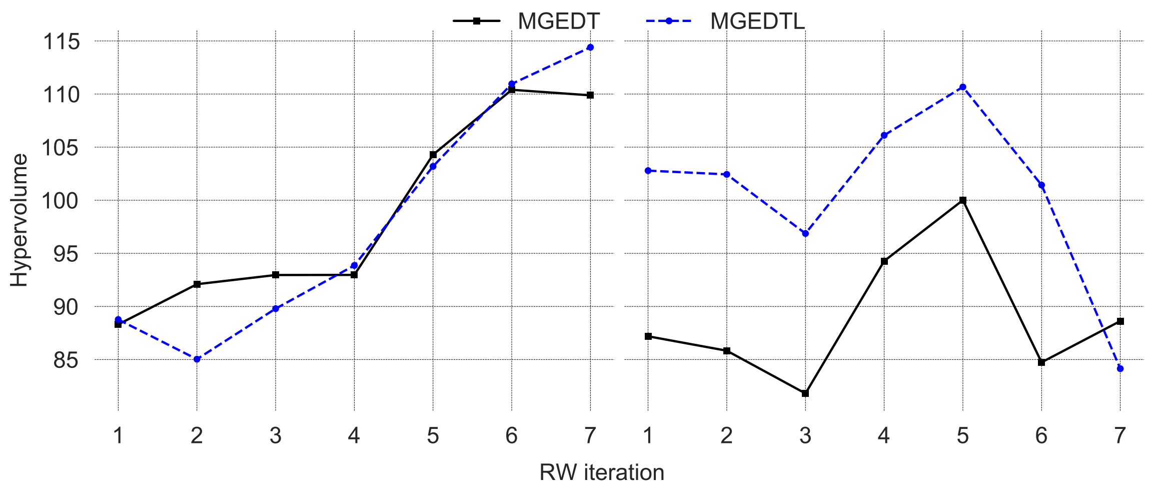


Figure 22: Evolution of the hypervolume values of the GE algorithms for BEST (left) and TEST (right) traffic (x-axis denotes the RW iteration and y-axis the AUC value).

Figure 23 plots the best -AUC fitness values that were evolved during the training GE generations for two initialization strategies and MGEDT: seeding, that reuses the best trees from the previous RW iteration; and no seeding, that performs a random population initialization. Each curve is computed as the average over the last six RW iterations (when there are dynamic time data updates). Figure 23 demonstrates the value of seeding, since this strategy produces better fitnesses during the whole evolution procedure.

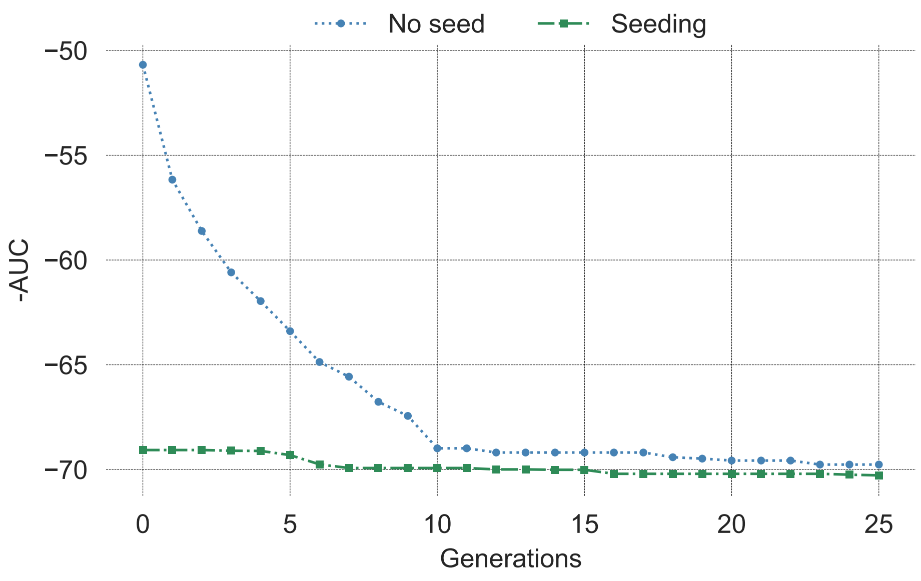


Figure 23: The effect of seeding a population (average of best AUC values for the last 6 RW iterations).

To demonstrate the quality of the GE predictive performance, Figure 24 details the respective ROC (left) and accumulated LIFT (right) curves that correspond to the best AUC trees that were obtained during the last RW iteration for the test (unseen) BEST traffic data. The ROC curves show a very good discrimination, with AUC values of 92.6% (MGEDTL) and 80.1% (MGEDT) that are much better than the random classifier (baseline, with AUC of 50%), confirming also that the predictive model performs well when faced with unbalanced test data. The LIFT curve is a popular analysis tool in the marketing domain (Moro et al., 2014). The accumulated curve plots the test samples sorted by the classifier probabilities (decreasing order) in the x-axis and the percentage of user positive responses (conversions) in the y-axis. The Area of the LIFT curve (ALIFT) shows how good the classifier distinguishes positive examples from all examples. Figure 24 shows that the very good AUC results correlate with similar ALIFT values (92.1% for MGEDTL) and (79.6% for MGEDT), which are very useful for the analyzed marketing domain. For instance, the LIFT curves show that if just 20% of the users were redirected to an advertisement (which would highly reduce the irritation of users that visit a publisher), then 90% and 60% of the conversions would be obtained by using the MGEDTL and MGEDT tree models.

In order to understand the computational effort, the training and testing times were monitored for all ML methods. Table 11 presents the average measured results (for all RW iterations) for each traffic

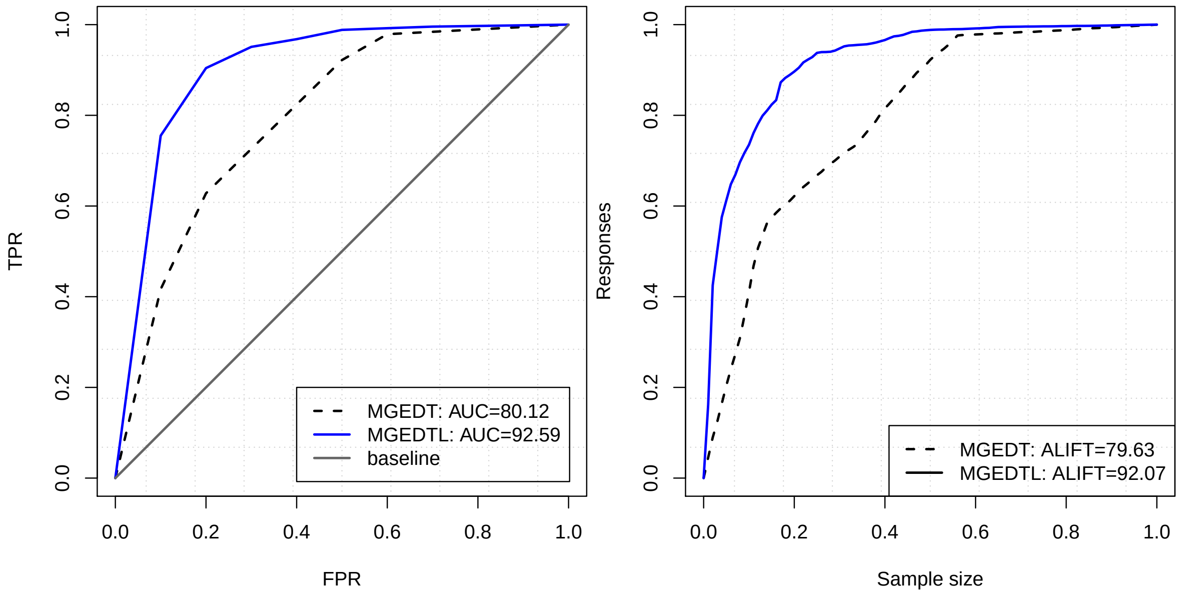


Figure 24: Example of ROC (left) and accumulated LIFT (right) curves for the GE algorithms and BEST traffic.

type in terms of: **Training** time (in s); **Prediction** time for one redirect (in μs); and **Campaigns**, the number of different campaigns that could be analyzed within the 10 ms (10,000 μs) DSP response time for matching an advertisement to a user when she/he clicks on a dynamic link. As expected, the faster training time method is provided by the standard DT, followed by the DL and RF. The GE methods require a higher training time, specially the Lamarckian variant (MGEDTL). Nevertheless, this algorithm is still affordable to be used in a real DSP environment, since it only requires a maximum training time of 116 minutes to be trained with two days of DSP data. And the GE training results could be substantially improved if a higher number of cores were used. For example, since the population size is 100 and 25 cores were used so far, the training time could be potentially reduced by a factor up to 4 if 100 cores were available. The most important constraint is the real-time DSP restriction for performing user-advertisement matches (10 ms limit). The prediction time results favor the GE and standard DT methods, with MGEDT obtaining the best predictive response times, followed by DT and MGEDTL. On average, the models evolved by MGEDT can be used to analyze the impressive number of 2 and 5 million campaigns, while MGEDTL evolved trees allow a selection among around 7 and 5 thousand campaigns. In contrast, DL (the slowest prediction time method) can only test around 200 campaigns, while RF (second slowest model) can handle around 500 campaigns. While there might be other computations performed by the DSP, these results are consistent with a real-time usage of the trees evolved by the GE methods, since there are typically around 400 (BEST) to 600 (TEST) active campaigns that need to be analyzed by the DSP.

The selection of the best approach is dependent on the Mobile Marketing DSP manager needs. The non GE approaches (DT, RF and DL) have the disadvantage of maximizing only one trade-off between predictive performance and model complexity (single point at Figure 19). In contrast, the GE methods

Table 11: Computational effort for all ML classifiers (average results using all RW iterations, best times in **bold**).

Mode	Model	Training (s)	Predict (μs)	Campaigns
BEST	MGEDT	893.13	**0.005**	2,000,000
	MGEDTL	6970.52	1.433	6,978
	DL	48.19	44.000	227
	DT	**8.98**	0.241	41,493
	RF	277.66	20.589	486
TEST	MGEDT	488.08	**0.002**	5,000,000
	MGEDTL	5662.30	2.032	4,921
	DL	34.72	56.000	178
	DT	**4.19**	0.236	42,372
	RF	132.64	18.358	545

perform a simultaneous optimization of both performance and ML complexity, obtaining a Pareto curve of ML models that provides more choices for the decision maker. If a high quality user conversion prediction model is needed, then the Lamarckian GE (MGEDTL) is an interesting option, since the evolved trees provide similar performances to the best DL/RF methods (Table 10) but much faster predictive responses, thus allowing to analyze more mobile campaigns (Table 11). In effect, Figure 19 confirms that MGEDTL evolves the least complex models for the Pareto trade-offs with AUC values ranging from around 80% to 92%. However, if very fast prediction or more interpretable models are needed (e.g., for design of new campaigns), then the pure GE algorithm is advised. In practice, both algorithms can be used by the marketing managers, who can select different trade-offs according to their real-time ML usage needs. Indeed, the GE results (predictive performance, complexity and computational effort) were shown to the DSP company managers, who provided a very positive feedback. In effect, the company plans to integrate the GE algorithms in their DSP platform.

4.4 Conclusions

The Mobile Performance Marketing industry value is increasing due to a worldwide usage of mobile devices (e.g., smartphone, tablet). It consists of markets supported by Demand-Side Platforms (DSP), which match advertisement to dynamic links activated by users. In these markets, monetary compensation only occurs when there is a product or service acquisition (the conversion). A crucial DSP expert system design issue is the nontrivial task of user Conversion Rate (CVR) prediction, that consists of selecting the best advertisements (with highest conversion probability) for each user, thus improving the overall CVR.

In this chapter, we address the user CVR prediction task, which involves several quality measures: the predictive performance of the Machine Learning (ML) classifier, the interpretability of the fitted model and real-time DSP advertisement match responses (e.g., under 10 ms). We consider recent real-world big data from a worldwide Mobile Marketing company (OLAmobile), which includes around 6 million of data events under two main traffic modes: TEST and BEST.

In terms of the ML classifiers, we propose a new Multi-objective Optimization (MO) approach to create and evolve Decision Trees (DT) using a Grammatical Evolution (GE). In particular, we explore two GE variants: MGEDT – a pure GE; and MGEDTL – a GE that uses a Lamarckian Evolution (LE), in which a local search method (CART algorithm) is applied to a randomly selected subtree and the improved solutions are encoded back into the GE genome. Both variants are capable of evolving DT with an arbitrary size and perform a simultaneous optimization of the predictive performance, measured by the AUC metric, and complexity, measured by the size of the genome tree (the coding region). Moreover, the two GE methods can cope with big data by using a sampling mechanism of training data and parallelism with multi-core processors.

To validate the proposed GE algorithms, a realistic Rolling Window (RW) evaluation was adopted, that included two days of training data and six hours of test data, a total of seven training and test iterations were executed. The two GE variants were compared with a standard DT (CART), a Random Forest and a state-of-the-art Deep Learning (DL) algorithm. Overall, competitive results were achieved by the GE methods. In effect, the MGEDTL obtaining the best predictive performance versus complexity trade-offs for the 80% to 92% AUC range, with around 10,000 GE genes in the coding region, while the MGEDT resulted in simpler trees for the range between 50% and 80%. As for the computational effort, both GE have affordable training times (e.g., 116 minutes when using around 1 million training records). More importantly, they evolve trees that have real-time prediction responses, where thousands of campaigns can be analysed by the best evolved trees. Depending on the Mobile Marketing manager needs, both GE methods can be used in a real DSP environment. If the pressure is set to increase user CVR, then MGEDTL is the best option, as it provides an excellent predictive performance (e.g., AUC of 92%). However, if interpretability (e.g., to help in the design new campaigns) or very fast prediction response times are valuable, then MGEDT provides better tree trade-offs (e.g., AUC of 80% and complexity tree with around 50 genes).

The GE results were shown to the analyzed DSP company, which found them very interesting. For instance, the current matching algorithm used by the DSP has a random performance (AUC of 50%) for the TEST data. The proposed GE methods provide a substantial predictive improvement. In effect, the highest performing trees result in a predictive performance that presents a 42 (MGEDTL) and a 30 (MGEDT) percentage point difference when compared with the DSP algorithm. Moreover, the results show that high quality AUC predictive models correlate with interesting accumulated LIFT curves. For example, during the last RW iteration for BEST traffic, the tree evolved by MGEDTL with the highest AUC (92.6%) produced an excellent LIFT (area of 92.1%), which indicates that it is possible to obtain

90% of the conversions if only 20% of the users were redirected to an advertisement. Thus, there is a potential to highly reduce the irritation of users that visit a publisher while only slightly reducing the profit. In effect, if a significantly lower, albeit more selective, number of advertisement matches were adopted then probably a higher audience would be captured by the publishers, which would result in a higher monetary compensation for the DSP players (DSP provider, publishers and advertisers).

An additional and important advantage of the proposed GE algorithms is related with the future deployment and maintenance of the DSP ML classifiers. Some ML algorithms require a nontrivial hyperparameter selection. For instance, a DL architecture is often set by ML experts, since it contains a large number of setup values, including the neural network topology configuration. Mobile Marketing data has high velocity and there are dynamic market and DSP changes, such as new campaigns, changes in the online buying behaviors and new DSP features. Since the proposed GE performs an automatic design of the DTs, it would be much easier for the DSP developers to continuously create new data-driven models, more adjusted to their needs.

In future work, we wish to test the proposed GE methods in a real DSP environment, tracking the quality of the evolved models during a longer time period. Indeed, the analysed DSP company plans to perform such an implementation. During this implementation, we believe it would be interesting to perform a code optimization, by making use of more efficient parallelization implementations (e.g., Apache Spark). Also, we intend to further validate the proposed GE approach to build generic DTs by analyzing classification datasets from distinct domains. Another interesting research direction is to adapt the MO GE approach to other classification tasks (e.g., multi-class, ordinal). Finally, we wish to test different representations with the GE, such as symbolic regression expressions or neural networks.

Part III

CONCLUSIONS

5

CONCLUSIONS

5.1 Overview

The advances in *Information and Communication Technology (ICT)*, associated to the easy access to mobile devices (e.g., smartphones), leveraged diverse digital marketing business opportunities, namely in the advertisement sector. In particular, Mobile Performance Marketing has been a target of vast investments over the last years (Statista, 2021). This market, also known as Performance-based Advertising, is related with the presentation of product and services campaigns in mobile devices. It involves 4 main stakeholders:

- **Advertisers** – entities that wish to sell mobile products, or offer service subscriptions. To do so, they rent digital spaces where their campaigns can be presented, only paying for it when there are measurable results.
- **Publishers** – owners of digital spaces, usually with a vast audience of users to whom campaigns can be presented to. They rent digital spaces to advertisers that will present their marketing campaigns by means of a smart link.
- **Users** – owners of mobile devices who are presented with advertisements whenever they are accessing interesting publisher services (e.g., surfing through websites or playing video games). In some cases, users are forced to click in the campaigns in order to continue accessing the desired content, being redirected to an ad campaign page.
- **Intermediary companies** – entities that act as brokers and that are responsible for matching users with interesting campaigns, with the purpose of leading them to perform a purchase or subscription (conversion). These companies often own *Demand-Side Platforms (DSPs)*, which are platforms responsible for automatically selecting an advertisement to be presented to users.

In Mobile Performance Marketing, the revenue flows from the user, when she/he buys or subscribes a product or service, to the advertiser. Then, a part of this revenue is automatically returned to the publisher by means of the DSP. The DSP also receives a small percentage of the revenue, in order to finance the intermediary companies. It is important to notice that this business represents a very low risk to the advertisers, since the compensation only occurs whenever a product is acquired or a service is subscribed. Therefore, improving the DSP performance when matching users to campaigns can have a direct impact in the profits for the three involved entities (advertisers, publishers and DSP companies).

This doctoral thesis was partially developed under a *Research and Development (R&D)* project termed *PRediction and Optimization of MObile Subscription marketing campaigns (PROMOS)*, born from a consortium between University of Minho and OLAmobile, an organisation that owns a DSP and operates in the Mobile Performance Marketing as an intermediary company. When this project started, back in 2017, OLAmobile specialists performed a market analysis with their partners and competitors, concluding that none of them presented the more interesting campaign to users. Instead, the advertisement selection was performed based on static business rules based on simple statistics (e.g., past performance of a campaign for a particular mobile operator and country). This approach results on a tiny percentage (nearly 1%) of user conversions. Therefore, the main goal of the PROMOS project was the improvement of mobile user *Conversion Rate (CVR)* rate, and consequently several other metrics associated with this industry, by applying *Artificial Intelligence (AI)* technologies, more specifically *Machine Learning (ML)*, to the process of matching users to campaigns.

Over the last years, ML technologies have been widely used in several domains, presenting highly successful results. However, traditional ML approaches present difficulties when handling vast data records. Furthermore, recent studies have shown the strength of combining Modern Optimization, particularly *Evolutionary Algorithms (EAs)*, to optimize different ML models, outperforming traditional approaches (Stanley and Miikkulainen, 2002; Mendes et al., 2002; Cortez and Donate, 2014; Donate and Cortez, 2014; Ojha et al., 2017; Pereira et al., 2017). These algorithms are easily adaptable to any specific problem (Cortez, 2021) and tend to provide good results with a reasonable computational cost (Michalewicz et al., 2006). Thus, during this PhD thesis we explored EAs for optimizing predictive models adapted for the described Mobile Performance Marketing user CVR task. In particular, we have considered multi-objective approaches, namely *Non-dominated Sorting Genetic Algorithm II (NSGA-II)*, allowing us to simultaneously optimize more than one goal.

This thesis main objective was to give a strong contribution to the body of knowledge of using Modern Optimization to automatically design predictive models. In particular, the analyzed Mobile Performance Marketing domain contains several ML challenging issues, namely: it involves big data, since typically millions of clicks are generated every hour; only a tiny amount of user clicks are converted into a sale; due to technological constrains and privacy issues (e.g., it is not possible to identify a single user), only a limited set of data features is available; all considered data features are categorical and several of these features present a large cardinality, with hundreds or thousands of levels. To handle this complex domain, we focused on multi-objective ML tasks, in order to achieve more flexible predictive models that could satisfy the Mobile Marketing needs. We note that at an initial stage of this PhD work, we did not have access to OLAmobile DSP data. Thus, we first explored a multi-objective Modern Optimization of a distinct ML task: the design of *Neural Networks (NNs)* for time series *Prediction Intervals (PIs)*. Later, once the DSP data was made available, we reused the gained multi-objective experience (e.g., usage of the same NSGA-II algorithm) to the main PhD task: the prediction of mobile user CVR. As explained in Chapter 1, a different base learner (*Decision Trees (DTs)*) was adopted during this step. The specific contributions of this thesis are:

- We propose 6 multi-objective neuroevolution *Multi-objective evolutionary algorithm Lower and Upper Bound Estimation (MLUBE)* methods for multi-step ahead time series PIs based on *Evolutionary Artificial Neural Network (EANN)*. These methods used the NSGA-II algorithm to minimize both the *Normalized Mean Prediction Interval Width (NMPIW)* and *Prediction Interval Coverage Error (PICE)* quality measures.

- We designed a novel robust evaluation method for multi-objective ML tasks that vertically aggregates similar solutions (based on a chosen metric) using Wilcoxon median values and 95% confidence intervals. This evaluation method allows an easy result comparison between several optimized multi-objective solutions, as achieved when using multiple *Rolling Window (RW)* iterations.

- We propose two multi-objective *Grammatical Evolution (GE)*-based approaches for designing and evolving DTs. One of them (*Multi-objective Grammatical Evolution Decision Tree (MGEDT)*) uses a pure GE method, while the other (*Multi-objective Grammatical Evolution Decision Tree Lamarckian evolved (MGEDTL)*) takes advantage of Lamarckian Evolution. Both of them are able to deal with high-speed unbalanced data, take advantage of multi-core processing, and were applied to Mobile Performance Marketing data. Furthermore, we performed an empirical comparison between the proposed GEs and several ML algorithms, namely DT, *Random Forest (RF)* and *Deep Learning (DL)*, using our novel multi-objective evaluation method.

5.2 Discussion And Future Work

During the first year of this PhD project, there was no available data regarding the Mobile Performance Marketing industry, since OLAmobile was still developing an *Application Programming Interface (API)* to perform a data collection process for the research purposes. In that period, we chose to start exploring a multi-objective optimization of EANN for PI, giving continuation of our previous work in the area of evolving ML predictive models. The purpose was to gain further insights and skills on this subject, which could be useful once OLAmobile data was collected and the final R&D PROMOS requirements were established. Our developed work led to one Q1 Journal article (Cortez et al., 2020) and one conference paper (Pereira et al., 2017), where we proposed different neuroevolution models and a novel and feasible evaluation method for multi-objective ML tasks, as described in Chapter 3.

While interesting results were obtained by the neuroevolution MLUBE methods, we did not directly apply these to the Mobile Performance Marketing domain. In effect, once we had access to OLAmobile business requirements and associated data, these methods felt outside the R&D PROMOS scope since time series forecasting could not directly impact on ad selection for a particular user. Nevertheless, in future research, the proposed neuroevolution models could be more easily adapted for regression tasks that could impact the Mobile Performance Marketing domain. For instance, by addressing the user CVR prediction goal as an ordinal classification, such as performed by Matos et al. (2019b), namely a five class prediction (”no sale”, ”very low”, ”low”, ”medium” or ”high”), or even as a pure regression task (0 if ”no sale”, else the value of the conversion).

In a later step of the PhD execution, once OLAmobile data was available, and considering that another PhD thesis related to the R&D PROMOS project was being developed using NNs, in particular DL architectures, we decided to change the base learner, i.e., the ML model to be designed and evolved by the EAs. Thus, we performed several preliminary experiments using *Genetic Programming (GP)* and GE to design and evolve Logistic Regression and Symbolic Regression models. The purpose was to predict if a conversion would occur once a user is redirected to a campaign page. While this goal was addressed as a binary classification task, the model output was a class probability, since it was considered more informative (e.g., to rank different campaign ads) and it also allows the estimation of the *Area Under the Curve (AUC)* of the *Receiver Operating Characteristic (ROC)* analysis, which measures the quality of a binary class discrimination prediction regardless of the data unbalancing. Recent studies proposed similar approaches to different domains, achieving interesting results (Nyathi and Pillay, 2018; Tzimourta et al., 2018; Nicolau and Agapitos, 2021). Yet, and despite having explored several modeling attempts, we were not able to achieve quality results with these base learners when applied to the Mobile Performance Marketing data. Due to the temporal restrictions of this PhD thesis and following on some recent state-of-the-art works, which applied EAs to evolve DTs (e.g., Chabbouh et al. (2019), Czajkowski and Kretowski (2019a)), we opted to change the target base learner, resulting in the work detailed in Chapter 4. Nevertheless, the usage of GE to evolve distinct learners, such as symbolic expressions (Azad and Ryan, 2018) or neural networks (de Lima and Pozo, 2019), is an interesting future research direction.

After changing our base learner to DTs, motivated by its popularity in classification tasks, capability to produce human understandable models and also building upon some recent similar works (e.g., (Chabbouh et al., 2019), (Czajkowski and Kretowski, 2019a)), we were able to provide interesting research contributions both to the scientific community and to the Performance Marketing Industry. In particular, we developed two GE-based approaches to design and evolve DTs, simultaneously maximizing its predictive performance and minimizing its complexity, as detailed in Chapter 4. Both approaches generate solutions able to deal with high-speed unbalanced data in online learning environments, taking advantage of multi-core processing techniques. Using our innovative evaluation method, we performed a benchmark empirical study, comparing our results with two state-of-the-art tree-based ML algorithms (DT and RF) and a DL architecture proposed by Matos et al. (2019a), leading to the core publication of this PhD thesis (Pereira et al., 2021). While we have applied our *Evolutionary Decision Trees (EDTs)* to a binary classification task, the obtained output is a probability associated with the chance of occurring a conversion, and this probability can be used to rank alternative ad campaigns. In effect, using the evolved DTs and assuming a 10 millisecond DSP maximum response time for a single dynamic ad click assignment, we can test from around 5,000 (with MGEDTL) to 2,000,000 (with MGEDT) different campaigns and select the one with the highest probability of conversion. These results were shown to the OLAmobile company, which provided a very positive feedback. In effect, the company is currently implementing the proposed EDT algorithms in their DSP platform. Moreover, for a broader usage of our EDTs, we have created a publicly available Python module with our implementations and that can be installed via Pypi[1] or GitHub[2] (Appendix A presents the user manual). In effect, we believe there is a huge potential for the usage of the proposed EDTs for other ML application domains that involve classification tasks (e.g., direct marketing, credit score assignment).

While reviewing the studies related with the Modern Optimization of predictive models, particularly in terms of EDTs, we found interesting approaches applied to different domains. We have chosen to evolve axis-parallel DTs, since these type of DT are more easily understandable. Nevertheless, it should be noted that there are some related works that obtained interesting results using different tree representations (e.g., oblique, mixed) (Chabbouh et al., 2019; Czajkowski and Kretowski, 2019b). Therefore, we intend to address these tree representations in future work. In addiction, we also wish to explore different parallelism approaches, namely the Island Model that already proved its advantages both in terms of speed and on the convergence of the evolutionary process (Corcoran and Wainwright, 1994; Whitley et al., 1998).

Nowadays business needs often involve dealing with two or more important goals. Despite the actual huge interest on the ML topic, particularly in DL, most ML prediction models cannot naturally handle conflicting multi-objective tasks. During this PhD thesis, we were able to identify and fulfill a gap in the Mobile Performance Marketing industry, also achieving relevant scientific contributions in the multi-objective Modern Optimization of predictive models (including the prediction of time series intervals). However, there is some scarcity of studies proposing Pareto-based solutions for multi-objective ML tasks, particularly concerning its use for tackling the complexity problem that is inherent from bloat problems. This work addresses this problem successfully and opens new research venues in this area.

1 https://pypi.org/project/evoltree/
2 https://github.com/p-pereira/evoltree

REFERENCES

Agarwal, D., Long, B., Traupman, J., Xin, D., and Zhang, L. (2014). LASER: a scalable response prediction platform for online advertising. In Carterette, B., Diaz, F., Castillo, C., and Metzler, D., editors, *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014*, pages 173–182. ACM.

Ahmadizar, F., Soltanian, K., AkhlaghianTab, F., and Tsoulos, I. (2015). Artificial neural network development by means of a novel combination of grammatical evolution and genetic algorithm. *Engineering Applications of Artificial Intelligence*, 39:1–13.

Ak, R., Li, Y., Vitelli, V., Zio, E., López Droguett, E., and Magno Couto Jacinto, C. (2013). NSGA-II-trained neural network approach to the estimation of prediction intervals of scale deposition rate in oil & gas equipment. *Expert Systems with Applications*, 40(4):1205–1212.

Ak, R., Vitelli, V., Zio, E., and Member, S. (2015). An Interval-Valued Neural Network Approach for Uncertainty Quantification in Short-Term Wind Speed Prediction. *IEEE Transactions on Neural Networks and Learning Systems*, 26(11):2787–2800.

Angeline, P. J., Saunders, G. M., and Pollack, J. B. (1994). An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 5(1):54–65.

Arrieta, A. B., Rodríguez, N. D., Ser, J. D., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. (2020). Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115.

Ashofteh, P.-S., Haddad, O. B., and Loáiciga, H. A. (2015). Evaluation of climatic-change impacts on multiobjective reservoir operation with multiobjective genetic programming. *Journal of Water Resources Planning and Management*, 141(11):04015030.

Assunção, F., Lourenço, N., Machado, P., and Ribeiro, B. (2017). Towards the evolution of multi-layered neural networks: A dynamic structured grammatical evolution approach. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, Berlin, Germany, July 15-19, 2017*, GECCO '17, page 393–400. Association for Computing Machinery.

Azad, R. M. A. and Ryan, C. (2018). Comparing methods to creating constants in grammatical evolution. In Ryan, C., O'Neill, M., and Collins, J. J., editors, *Handbook of Grammatical Evolution*, pages 245–262. Springer.

Barros, R. C., Basgalupp, M. P., de Carvalho, A. C. P. d. L. F., and Freitas, A. A. (2012). A survey of evolutionary algorithms for decision-tree induction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 42(3):291–312.

Barros, R. C., Ruiz, D., and Basgalupp, M. P. (2011). Evolutionary model trees for handling continuous classes in machine learning. *Information Sciences*, 181(5):954–971.

Bartoli, A., Castelli, M., and Medvet, E. (2020). Weighted hierarchical grammatical evolution. *IEEE Transactions on Cybernetics*, 50(2):476–488.

Battiti, R. and Brunato, M. (2017). *The LION way. Machine Learning* plus *Intelligent Optimization*. LIONlab, University of Trento, Italy.

Beazley, D. (2001). Ply (python lex-yacc). http://www.dabeaz.com/ply.

Ben Chaabene, W. and Nehdi, M. L. (2021). Genetic programming based symbolic regression for shear capacity prediction of sfrc beams. *Construction and Building Materials*, 280:122523.

Beume, N., Fonseca, C. M., López-Ibáñez, M., Paquete, L., and Vahrenhold, J. (2009). On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation*, 13(5):1075–1082.

Bianchini, M. and Scarselli, F. (2014). On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 25(8):1553–1565.

Bianco, S., Ciocca, G., and Schettini, R. (2017). Combination of video change detection algorithms by genetic programming. *IEEE Transactions on Evolutionary Computation*, 21(6):914–928.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Brownlee, J. (2011). *Clever Algorithms: Nature-Inspired Programming Recipes*. Lulu.com, 1st edition.

Budinich, M. (1996). A self-organizing neural network for the traveling salesman problem that is competitive with simulated annealing. *Neural Computation*, 8(2):416–424.

Campos, G. O., Zimek, A., Sander, J., Campello, R. J. G. B., Micenková, B., Schubert, E., Assent, I., and Houle, M. E. (2016). On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 30(4):891–927.

Cardamone, L., Loiacono, D., and Lanzi, P. L. (2010). Learning to drive in the open racing car simulator using online neuroevolution. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(3):176–190.

Chabbouh, M., Bechikh, S., Hung, C., and Said, L. B. (2019). Multi-objective evolution of oblique decision trees for imbalanced data binary classification. *Swarm and Evolutionary Computation*, 49:1–22.

Chandra, R. and Chand, S. (2016). Evaluation of co-evolutionary neural network architectures for time series prediction with mobile application in finance. *Applied Soft Computing*, 49:462–473.

Chatfield, C. (2000). *Time-series forecasting*. CRC Press.

Chen, Q., Xue, B., and Zhang, M. (2019). Improving generalization of genetic programming for symbolic regression with angle-driven geometric semantic operators. *IEEE Transactions on Evolutionary Computation*, 23(3):488–502.

Chen, Y., Pavlov, D., and Canny, J. F. (2009). Large-scale behavioral targeting. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 209–218. ACM.

Chryssolouris, G., Lee, M., and Ramsey, A. (1996). Confidence interval prediction for neural network models. *IEEE Transactions on Neural Networks and Learning Systems*, 7(1):229–232.

Colmenar, J. M., Risco-Martín, J. L., Atienza, D., and Hidalgo, J. I. (2011). Multi-objective optimization of dynamic memory managers using grammatical evolution. In Krasnogor, N. and Lanzi, P. L., editors, *13th Annual Genetic and Evolutionary Computation Conference, GECCO 2011, Proceedings, Dublin, Ireland, July 12-16, 2011*, pages 1819–1826. ACM.

Corcoran, A. L. and Wainwright, R. L. (1994). A parallel island model genetic algorithm for the multiprocessor scheduling problem. In Berghel, H., Hlengl, T., and Urban, J. E., editors, *Proceedings of the 1994 ACM Symposium on Applied Computing, SAC'94, Phoenix, AZ, USA, March 6-8, 1994*, pages 483–487. ACM.

Cortez, P. (2021). *Modern Optimization with R*. Springer.

Cortez, P. and Donate, J. P. (2014). Global and decomposition evolutionary support vector machine approaches for time series forecasting. *Neural Computing and Applications*, 25(5):1053–1062.

Cortez, P. and Embrechts, M. J. (2013). Using sensitivity analysis and visualization techniques to open black box data mining models. *Information Sciences*, 225:1–17.

Cortez, P., Matos, L. M., Pereira, P. J., Santos, N., and Duque, D. (2016). Forecasting store foot traffic using facial recognition, time series and support vector machines. In *International Joint Conference SOCO'16-CISIS'16-ICEUTE'16 - San Sebastián, Spain, October, 2016*, volume 527 of *Advances in Intelligent Systems and Computing*, pages 267–276.

Cortez, P., Pereira, P. J., and Mendes, R. (2020). Multi-step time series prediction intervals using neuroevolution. *Neural Computing and Applications*, 32(13):8939–8953.

Cortez, P., Rio, M., Rocha, M., and Sousa, P. (2012). Multi-scale internet traffic forecasting using neural networks and time series methods. *Expert Systems*, 29(2):143–155.

Cortez, P., Rocha, M., and Neves, J. (2002). A lamarckian approach for neural network training. *Neural Processing Letters*, 15(2):105–116.

Czajkowski, M. and Kretowski, M. (2010). Globally induced model trees: An evolutionary approach. In Schaefer, R., Cotta, C., Kolodziej, J., and Rudolph, G., editors, *Parallel Problem Solving from Nature - PPSN XI, 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I*, volume 6238, pages 324–333. Springer LNCS.

Czajkowski, M. and Kretowski, M. (2013). Global induction of oblique model trees: An evolutionary approach. In Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L. A., and Zurada, J. M., editors, *Artificial Intelligence and Soft Computing - 12th International Conference, ICAISC 2013, Zakopane, Poland, June 9-13, 2013, Proceedings, Part II*, volume 7895, pages 1–11. Springer LNCS.

Czajkowski, M. and Kretowski, M. (2019a). Decision tree underfitting in mining of gene expression data. an evolutionary multi-test tree approach. *Expert Systems with Applications*, 137:392–404.

Czajkowski, M. and Kretowski, M. (2019b). A multi-objective evolutionary approach to pareto-optimal model trees. *Soft Computing*, 23(5):1423–1437.

de Lima, R. H. R. and Pozo, A. T. R. (2019). Evolving convolutional neural networks through grammatical evolution. In López-Ibáñez, M., Auger, A., and Stützle, T., editors, *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2019, Prague, Czech Republic, July 13-17, 2019*, pages 179–180. ACM.

Deng, L., Gao, J., and Vuppalapati, C. (2015). Building a big data analytics service framework for mobile advertising and marketing. In *First IEEE International Conference on Big Data Computing Service and Applications, BigDataService 2015, Redwood City, CA, USA, March 30 - April 2, 2015*, pages 256–266. IEEE Computer Society.

Dhaenens, C. and Jourdan, L. (2016). *Metaheuristics for Big Data*. Computer Engineering Series: Metaheuristics Set. John Wiley & Sons, Ltd.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In Kittler, J. and Roli, F., editors, *Multiple Classifier Systems, First International Workshop, MCS 2000, Cagliari, Italy, June 21-23, 2000, Proceedings*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer.

Doerr, B., Kötzing, T., Lagodzinski, J. A. G., and Lengler, J. (2017). Bounding bloat in genetic programming. In Bosman, P. A. N., editor, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, Berlin, Germany, July 15-19, 2017*, pages 921–928. ACM.

Domingos, P. M. and Hulten, G. (2000). Mining high-speed data streams. In Ramakrishnan, R., Stolfo, S. J., Bayardo, R. J., and Parsa, I., editors, *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000*, pages 71–80. ACM.

Domo (2021). Data never sleeps 8.0.

Donate, J. P. and Cortez, P. (2014). Evolutionary optimization of sparsely connected and time-lagged neural networks for time series forecasting. *Applied Soft Computing*, 23:432–443.

Du, M., Sassioui, R., Varisteas, G., State, R., Brorsson, M., and Cherkaoui, O. (2017). Improving real-time bidding using a constrained markov decision process. In Cong, G., Peng, W., Zhang, W. E., Li, C., and Sun, A., editors, *Advanced Data Mining and Applications - 13th International Conference, ADMA 2017, Singapore, November 5-6, 2017, Proceedings*, volume 10604 of *Lecture Notes in Computer Science*, pages 711–726. Springer.

Du, M., State, R., Brorsson, M., and Avanesov, T. (2016). Behavior profiling for mobile advertising. In *Proceedings of the 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, BDCAT 2016, Shanghai, China, December 6-9, 2016*, pages 302–307. ACM.

Dybowski, R. and Roberts, S. (2000). Confidence intervals and prediction intervals for feed-forward neural networks. *Clinical Applications of Artificial Neural Networks*, pages 298–326.

Eiben, A. E. and Smith, J. E. (2015). *Introduction to Evolutionary Computing, Second Edition*. Natural Computing Series. Springer.

Fan, J., Upadhye, S., and Worster, A. (2006). Understanding receiver operating characteristic (roc) curves. *Canadian Journal of Emergency Medicine*, 8(1):19–20.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874.

Fenton, M., McDermott, J., Fagan, D., Forstenlechner, S., Hemberg, E., and O'Neill, M. (2017). Ponyge2: grammatical evolution in python. In *Genetic and Evolutionary Computation Conference, Berlin, Germany, July 15-19, 2017, Companion Material Proceedings*, pages 1194–1201. ACM.

Ferreira, C. (2001). Gene expression programming: A new adaptive algorithm for solving problems. *Complex Systems*, 13(2).

Fitzgerald, J. M., Azad, R. M. A., and Ryan, C. (2015). GEML: evolutionary unsupervised and semi-supervised learning of multi-class classification with grammatical evolution. In *Proceedings of the 7th International Joint Conference on Computational Intelligence (IJCCI 2015) - Volume 1: ECTA, Lisbon, Portugal, November 12-14, 2015*, pages 83–94. SciTePress.

Floreano, D., Dürr, P., and Mattiussi, C. (2008). Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62.

Furtuna, R., Curteanu, S., and Leon, F. (2012). Multi-objective optimization of a stacked neural network using an evolutionary hyper-heuristic. *Applied Soft Computing*, 12(1):133–144.

Gaber, M. M., Zaslavsky, A. B., and Krishnaswamy, S. (2005). Mining data streams: a review. *SIGMOD Record*, 34(2):18–26.

Gartner Research (2012). IT Glossary.

Glass, L. and Mackey, M. (1977). Oscillation and chaos in physiological control systems. *Science*, 197:287–289.

Gonçalves, S., Cortez, P., and Moro, S. (2020). A deep learning classifier for sentence classification in biomedical and computer science abstracts. *Neural Computing and Applications*, 32(11):6793–6807.

Gruau, F. (1994). Neural network synthesis using cellular encoding and the genetic algorithm.

Guijo-Rubio, D., Durán-Rosal, A., Gutiérrez, P., Gómez-Orellana, A., Casanova-Mateo, C., Sanz-Justo, J., Salcedo-Sanz, S., and Hervás-Martínez, C. (2020). Evolutionary artificial neural networks for accurate solar radiation prediction. *Energy*, 210:118374.

Guogis, E. and Misevicius, A. (2014). Comparison of genetic programming, grammatical evolution and gene expression programming techniques. In Dregvaite, G. and Damasevicius, R., editors, *Information and Software Technologies - 20th International Conference, ICIST 2014, Druskininkai, Lithuania, October 9-10, 2014. Proceedings*, volume 465, pages 182–193. Springer CCIS.

Han, J., Kamber, M., and Pei, J. (2011). *Data Mining: Concepts and Techniques, 3rd edition*. Morgan Kaufmann.

Hand, D., Mannila, H., and Smyth, P. (2001). *Principles of Data Mining Cambridge*, volume 2001. Springer, London.

Hand, D. J. (2006). Classifier technology and the illusion of progress. *Statistical science*, pages 1–14.

Hastie, T., Tibshirani, R., and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer Series in Statistics. Springer.

Hastings, E. J., Guha, R. K., and Stanley, K. O. (2009). Automatic content generation in the galactic arms race video game. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(4):245–263.

Hemberg, E., Kelly, J., and O'Reilly, U. (2019). On domain knowledge and novelty to improve program synthesis performance with grammatical evolution. In Auger, A. and Stützle, T., editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019, Prague, Czech Republic, July 13-17, 2019*, pages 1039–1046. ACM.

Heskes, T. (1996). Practical confidence and prediction intervals. In Mozer, M., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996*, pages 176–182. MIT Press.

Hollander, M., Wolfe, D. A., and Chicken, E. (2013). *Nonparametric statistical methods*. John Wiley & Sons.

Holst, A. (2021). Amount of data created, consumed, and stored 2010-2025.

Hyndman, R. (January, 2010). *Time Series Data Library*. http://robjhyndman.com/TSDL/.

Iqbal, M., Xue, B., Al-Sahaf, H., and Zhang, M. (2017). Cross-domain reuse of extracted knowledge in genetic programming for image classification. *IEEE Transactions on Evolutionary Computation*, 21(4):569–587.

Jankowski, D. and Jackowski, K. (2014). Evolutionary algorithm for decision tree induction. In *Computer Information Systems and Industrial Management - 13th IFIP TC8 International Conference, CISIM 2014, Ho Chi Minh City, Vietnam, November 5-7, 2014. Proceedings*, volume 8838 of *Lecture Notes in Computer Science*, pages 23–32. Springer LNCS.

Juárez-Smith, P., Trujillo, L., Valdez, M. G., de Vega, F. F., and de la O, F. C. (2019). Local search in speciation-based bloat control for genetic programming. *Genetic Programming and Evolvable Machines*, 20(3):351–384.

Kaur, M. and Singh, D. (2021). Multi-modality medical image fusion technique using multi-objective differential evolution based deep neural networks. *Journal of Ambient Intelligence and Humanized Computing*, 12(2):2483–2493.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of International Conference on Neural Networks (ICNN'95), Perth, WA, Australia, November 27 - December 1, 1995*, pages 1942–1948. IEEE.

Kessentini, M. and Ouni, A. (2017). Detecting android smells using multi-objective genetic programming. In *4th IEEE/ACM International Conference on Mobile Software Engineering and Systems, MOBILESoft@ICSE 2017, Buenos Aires, Argentina, May 22-23, 2017*, pages 122–132. IEEE.

Khoshgoftaar, T. M., Liu, Y., and Seliya, N. (2003). Genetic programming-based decision trees for software quality classification. In *15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2003), 3-5 November 2003, Sacramento, California, USA*, pages 374–383. IEEE Computer Society.

Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. (2011). Lower upper bound estimation method for construction of neural network-based prediction intervals. *IEEE Transactions on Neural Networks*, 22(3):337–346.

King, M. A., Abrahams, A. S., and Ragsdale, C. T. (2015). Ensemble learning methods for pay-per-click campaign management. *Expert Systems with Applications*, 42(10):4818–4829.

Kohl, N. and Miikkulainen, R. (2009). Evolving neural networks for strategic decision-making problems. *Neural Networks*, 22(3):326–337.

Koza, J. R. (1990). Concept formation and decision tree induction using the genetic programming paradigm. In Schwefel, H. and Männer, R., editors, *Parallel Problem Solving from Nature, 1st Workshop, PPSN I, Dortmund, Germany, October 1-3, 1990, Proceedings*, volume 496 of *Lecture Notes in Computer Science*, pages 124–128. Springer.

Koza, J. R. (1993). *Genetic programming - on the programming of computers by means of natural selection*. Complex adaptive systems. MIT Press.

Krempl, G., Zliobaite, I., Brzezinski, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., and Stefanowski, J. (2014). Open challenges for data stream mining research. *SIGKDD Explorations*, 16(1):1–10.

Kristo, T. and Maulidevi, N. U. (2016). Deduction of fighting game countermeasures using neuroevolution of augmenting topologies. In *2016 International Conference on Data and Software Engineering (ICoDSE)*, pages 1–6.

Lee, W. (2006). Genetic programming decision tree for bankruptcy prediction. In *Proceedings of the 2006 Joint Conference on Information Sciences, JCIS 2006, Kaohsiung, Taiwan, ROC, October 8-11, 2006*. Atlantis Press.

Leskovec, J., Rajaraman, A., and Ullman, J. D. (2014). *Mining of Massive Datasets, 2nd Ed.* Cambridge University Press.

Liang, J., Liu, Y., and Xue, Y. (2020). Preference-driven pareto front exploitation for bloat control in genetic programming. *Applied Soft Computing*, 92:106254.

Loshin, D. (2013). *Big Data Analytics*. Morgan Kaufmann.

Loveard, T. and Ciesielski, V. (2006). Representing classification problems in genetic programming. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, volume 2, pages 1070–1077 vol. 2.

Lowell, J., Grabkovsky, S., and Birger, K. (2011). Comparison of NEAT and hyperneat performance on a strategic decision-making problem. In Watada, J., Chung, P., Lin, J., Shieh, C., and Pan, J., editors, *Fifth International Conference on Genetic and Evolutionary Computing, ICGEC 2011, Kinmen, Taiwan / Xiamen, China, August 29 - September 1, 2011*, pages 102–105. IEEE Computer Society.

Luke, S. (2013). *Essentials of Metaheuristics (Second Edition)*. Lulu.com.

MacKay, D. J. C. (1992). The evidence framework applied to classification networks. *Neural Computation*, 4(5):720–736.

Makridakis, S., Weelwright, S., and Hyndman, R. (1998). *Forecasting: Methods and Applications*. John Wiley & Sons, New York, USA, third edition.

Mason, K., Duggan, J., and Howley, E. (2018). Forecasting energy demand, wind generation and carbon dioxide emissions in ireland using evolutionary neural networks. *Energy*, 155(16):705–720.

Matos, L. M., Cortez, P., Mendes, R., and Moreau, A. (2018). A comparison of data-driven approaches for mobile marketing user conversion prediction. In *9th IEEE International Conference on Intelligent Systems, IS 2018, Funchal, Madeira, Portugal, September 25-27, 2018*, pages 140–146. IEEE.

Matos, L. M., Cortez, P., Mendes, R., and Moreau, A. (2019a). Using deep learning for mobile marketing user conversion prediction. In *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*, pages 1–8. IEEE.

Matos, L. M., Cortez, P., Mendes, R. C., and Moreau, A. (2019b). Using deep learning for ordinal classification of mobile marketing user conversion. In *Intelligent Data Engineering and Automated Learning - IDEAL 2019 - 20th International Conference, Manchester, UK, November 14-16, 2019, Proceedings, Part I*, volume 11871 of *Lecture Notes in Computer Science*, pages 60–67. Springer.

Mehr, A. D. and Nourani, V. (2017). A pareto-optimal moving average-multigene genetic programming model for rainfall-runoff modelling. *Environmental Modelling and Software*, 92:239–251.

Mendes, R., Cortez, P., Rocha, M., and Neves, J. (2002). Particle swarms for feedforward neural network training. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, volume 2, pages 1895–1899 vol.2.

Mendes, R., Kennedy, J., and Neves, J. (2004). The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, 8(3):204–210.

Michalewicz, Z., Schmidt, M., Michalewicz, M., and Chiriac, C. (2006). *Adaptive Business Intelligence*. Springer-Verlag Berlin Heidelberg.

Mingo, J. M., Aler, R., Maravall, D., and de Lope Asiaín, J. (2013). Investigations into lamarckism, baldwinism and local search in grammatical evolution guided by reinforcement. *Computing and Informatics*, 32(3):595–627.

Morabito, V. (2015). *Big Data and Analytics: Strategic and Organizational Impacts*. Springer International Publishing.

Moro, S., Cortez, P., and Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31.

Moro, S., Cortez, P., and Rita, P. (2015). Using customer lifetime value and neural networks to improve the prediction of bank deposit subscription in telemarketing campaigns. *Neural Computing and Applications*, 26(1):131–139.

Motsinger-Reif, A. A., Deodhar, S., Winham, S. J., and Hardison, N. E. (2010). Grammatical evolution decision trees for detecting gene-gene interactions. *BioData Mining*, 3:8.

Mühlenbein, H. and Paass, G. (1996). From recombination of genes to the estimation of distributions i. binary parameters. In Voigt, H., Ebeling, W., Rechenberg, I., and Schwefel, H., editors, *Parallel Problem Solving from Nature - PPSN IV, International Conference on Evolutionary Computation. The 4th International Conference on Parallel Problem Solving from Nature, Berlin, Germany, September 22-26, 1996, Proceedings*, volume 1141 of *Lecture Notes in Computer Science*, pages 178–187. Springer.

Nettleton, D. (2014). *Commercial Data Mining: Processing, Analysis and Modeling for Predictive Analytics Projects*. Morgan Kaufmann.

Neukart, F. (2017). *Reverse Engineering the Mind: Consciously Acting Machines and Accelerated Evolution*. Springer.

Nicolau, M. and Agapitos, A. (2021). Choosing function sets with better generalisation performance for symbolic regression models. *Genetic Programming and Evolvable Machines*, 22(1):73–100.

North, M. (2012). *Data Mining for the Masses*. Global Text Project.

Nyathi, T. and Pillay, N. (2018). Comparison of a genetic algorithm to grammatical evolution for automated design of genetic programming classification algorithms. *Expert Systems with Applications*, 104:213–234.

Ojha, V. K., Abraham, A., and Snásel, V. (2017). Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, 60:97–116.

Oliveira, N., Cortez, P., and Areal, N. (2017). The impact of microblogging data for stock market prediction: Using twitter to predict returns, volatility, trading volume and survey sentiment indices. *Expert Systems with Applications*, 73:125–144.

O'Neill, M. and Brabazon, A. (2019). Mutational robustness and structural complexity in grammatical evolution. In *IEEE Congress on Evolutionary Computation, CEC 2019, Wellington, New Zealand, June 10-13, 2019*, pages 1338–1344. IEEE.

O'Neill, M. and Ryan, C. (2001). Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358.

O'Neill, M. and Ryan, C. (2003). *Grammatical evolution - evolutionary automatic programming in an arbitrary language*, volume 4 of *Genetic programming*. Kluwer.

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

Peralta Donate, J. and Cortez, P. (2014). Evolutionary optimization of sparsely connected and time-lagged neural networks for time series forecasting. *Applied Soft Computing*, 23:432–443.

Pereira, P. J., Cortez, P., and Mendes, R. (2017). Multi-objective learning of neural network time series prediction intervals. In *Progress in Artificial Intelligence - 18th EPIA Conference on Artificial Intelligence, EPIA 2017, Porto, Portugal, September 5-8, 2017, Proceedings*, volume 10423 of *Lecture Notes in Computer Science*, pages 561–572. Springer.

Pereira, P. J., Cortez, P., and Mendes, R. (2021). Multi-objective grammatical evolution of decision trees for mobile marketing user conversion prediction. *Expert Systems with Applications*, 168:114287.

Pereira, P. J., Pinto, P., Mendes, R., Cortez, P., and Moreau, A. (2019). Using neuroevolution for predicting mobile marketing conversion. In *Progress in Artificial Intelligence, 19th EPIA Conference on Artificial Intelligence, EPIA 2019, Vila Real, Portugal, September 3-6, 2019, Proceedings, Part II*, volume 11805 of *Lecture Notes in Computer Science*, pages 373–384. Springer.

Pham, S., Zhang, K., Phan, T., Ding, J., and Dancy, C. L. (2018). Playing SNES games with neuroevolution of augmenting topologies. In McIlraith, S. A. and Weinberger, K. Q., editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 8129–8130. AAAI Press.

Pradhan, M., Bamnote, G., Tribhuvan, V., Jadhav, K., Chabukswar, V., and Dhobale, V. (2012). A genetic programming approach for detection of diabetes. *International Journal of Computational Engineering Research*, 2(6):91–94.

Quiroz-Ramírez, O., Espinal, A., Ornelas-Rodríguez, M., Domínguez, A. R., Sánchez, D., Soberanes, H. J. P., Carpio, M., Mancilla-Espinoza, L. E., and Ortíz-López, J. (2018). Partially-connected artificial neural networks developed by grammatical evolution for pattern recognition problems. In Castillo, O., Melin, P., and Kacprzyk, J., editors, *Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications*, volume 749 of *Studies in Computational Intelligence*, pages 99–112. Springer.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rana, M., Koprinska, I., Khosravi, A., and Agelidis, V. G. (2013). Prediction intervals for electricity load forecasting using neural networks. In *The 2013 International Joint Conference on Neural Networks, IJCNN 2013, Dallas, TX, USA, August 4-9, 2013*, pages 1–8. IEEE.

Rao, S. S. (2019). *Engineering Optimization: Theory and Practice*. John Wiley & Sons, Ltd.

Rere, L. R., Fanany, M. I., and Arymurthy, A. M. (2015). Simulated annealing algorithm for deep learning. *Procedia Computer Science*, 72:137 – 144.

Rivera-López, R. and Canul-Reich, J. (2018). Construction of near-optimal axis-parallel decision trees using a differential-evolution-based approach. *IEEE Access*, 6:5548–5563.

Rocha, M., Cortez, P., and Neves, J. (2004). Evolutionary neural network learning algorithms for changing environments. *World Scientific and Engineering Academy and Society Transactions on Systems*, 3(2):596–601.

Ruiz, L. G. B., Delgado, R. R., Cuéllar, M. P., and del Carmen Pegalajar, M. (2018). Energy consumption forecasting based on elman neural networks with evolutive optimization. *Expert Systems with Applications*, 92:380–389.

Ryan, C., Collins, J. J., and O'Neill, M. (1998). Grammatical evolution: Evolving programs for an arbitrary language. In Banzhaf, W., Poli, R., Schoenauer, M., and Fogarty, T. C., editors, *Genetic Programming, First European Workshop, EuroGP'98, Paris, France, April 14-15, 1998, Proceedings*, volume 1391 of *Lecture Notes in Computer Science*, pages 83–96. Springer.

Sammut, C. and Webb, G. I., editors (2017). *Encyclopedia of Machine Learning and Data Mining*. Springer.

Santos, L. A. (2016). Neural Networks.

Schrum, J. and Miikkulainen, R. (2014). Evolving multimodal behavior with modular neural networks in ms. pac-man. In Arnold, D. V., editor, *Genetic and Evolutionary Computation Conference, GECCO '14, Vancouver, BC, Canada, July 12-16, 2014*, pages 325–332. ACM.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423.

Sotelo-Figueroa, M. A., Aguirre, A. H., Espinal, A., Soria-Alcaraz, J. A., and Ortíz-López, J. (2018). Symbolic regression by means of grammatical evolution with estimation distribution algorithms as search engine. In Castillo, O., Melin, P., and Kacprzyk, J., editors, *Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications*, volume 749 of *Studies in Computational Intelligence*, pages 169–177. Springer.

Soule, T. and Foster, J. A. (1998). Effects of code growth and parsimony pressure on populations in genetic programming. *Evolutionary Computation*, 6(4):293–309.

Srinivas, N. and Deb, K. (1994). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248.

Stanley, K. O. (2017). Neuroevolution: A different kind of deep learning.

Stanley, K. O., D'Ambrosio, D. B., and Gauci, J. (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212.

Stanley, K. O., Karpov, I., Miikkulainen, R., and Gold, A. (2006). Real-time interactive learning in the NERO video game. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 1953–1954. AAAI Press.

Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural network through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.

Statista (2021). Mobile advertising spending worldwide from 2007 to 2022.

Stepnicka, M., Cortez, P., Donate, J. P., and Stepnicková, L. (2013). Forecasting seasonal time series with computational intelligence: On recent methods and the potential of their combinations. *Expert Systems with Applications*, 40(6):1981–1992.

Sudholt, D. (2015). Parallel evolutionary algorithms. In *Springer Handbook of Computational Intelligence*, pages 929–959. Springer.

Suganuma, M., Shirakawa, S., and Nagao, T. (2017). A genetic programming approach to designing convolutional neural network architectures. In Bosman, P. A. N., editor, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, Berlin, Germany, July 15-19, 2017*, pages 497–504. ACM.

Sugiyama, M. (2016). *Biography*. Elsevier Science.

Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: an analysis and review. *International Forecasting Journal*, 16(4):437–450.

Tirumala, S. S. (2020). Evolving deep neural networks using coevolutionary algorithms with multi-population strategy. *Neural Computing and Applications*, 32(16):13051–13064.

Torgo, L. (2010). *Data Mining with R: Learning with Case Studies*. Chapman and Hall/CRC Press.

Trawinski, B. (2013). Evolutionary fuzzy system ensemble approach to model real estate market based on data stream exploration. *The Journal of Universal Computer Science*, 19(4):539–562.

Tsakonas, A., Dounias, G., Doumpos, M., and Zopounidis, C. (2006). Bankruptcy prediction with neural logic networks by means of grammar-guided genetic programming. *Expert Systems with Applications*, 30(3):449–461.

Tsoulos, I. G., Gavrilis, D., and Glavas, E. (2008). Neural network construction and training using grammatical evolution. *Neurocomputing*, 72(1-3):269–277.

Turban, E., Sharda, R., Delen, D., and King, D. (2010). *Business Intelligence: A Managerial Approach*. Prentice Hall.

Tzimourta, K., Tsoulos, I., Bilero, T., Tzallas, A., Tsipouras, M., and Giannakeas, N. (2018). Direct assessment of alcohol consumption in mental state using brain computer interfaces and grammatical evolution. *Inventions*, 3:51.

Vaishnavi, V. and Kuechler, B. (2004). *Design Research in Information Systems*, volume 22. Springer US.

Verbancsics, P. and Harguess, J. (2015). Image classification using generative neuro evolution for deep learning. In *2015 IEEE Winter Conference on Applications of Computer Vision, WACV 2015, Waikoloa, HI, USA, January 5-9, 2015*, pages 488–493. IEEE Computer Society.

Volna, E. (2010). *Introduction To Soft Computing*. Bookboon.

Whitley, D., Rana, S., and Heckendorn, R. B. (1998). The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7:33–47.

Witten, I. H., Frank, E., and Hall, M. A. (2011). *Data mining: practical machine learning tools and techniques, 3rd Edition*. Morgan Kaufmann, Elsevier.

Yu, L., Xu, H., and Tang, L. (2017). Lssvr ensemble learning with uncertain parameters for crude oil price forecasting. *Applied Soft Computing*, 56:692–701.

Yuan, S., Abidin, A. Z., Sloan, M., and Wang, J. (2012). Internet advertising: An interplay among advertisers, online publishers, ad exchanges and web users. *Computing Research Repository*, abs/1206.1754.

Zameer, A., Arshad, J., Khan, A., and Raja, M. A. Z. (2017). Intelligent and robust prediction of short term wind power using genetic programming based ensemble of neural networks. *Energy Conversion and Management*, 134:361–372.

Zhang, J., Zhang, J., Lok, T., and Lyu, M. R. (2007). A hybrid particle swarm optimization-back-propagation algorithm for feedforward neural network training. *Applied Mathematics and Computation*, 185(2):1026–1037.

Zhang, W., Yuan, S., and Wang, J. (2014). Real-time bidding benchmarking with ipinyou dataset. *CoRR*, abs/1407.7073.

Zhao, H. (2007). A multi-objective genetic programming approach to developing pareto optimal decision trees. *Decision Support Systems*, 43(3):809–826.

Zhong, J., Feng, L., Cai, W., and Ong, Y. (2020). Multifactorial genetic programming for symbolic regression problems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(11):4492–4505.

Part IV

APPENDICES

A

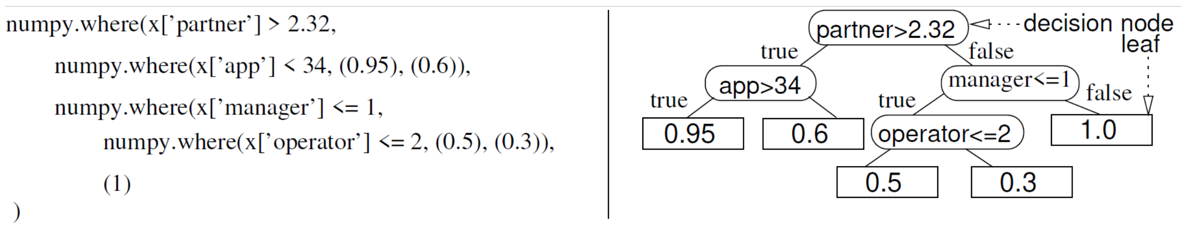
EVOLTREE: USER MANUAL

a.1 Overview

evoltree is a novel **Multi-objective Optimization (MO)** approach to evolve **Decision Trees (DT)** using **Grammatical Evolution (GE)**, under two main variants: a pure GE method (**EDT**) and a GE with Lamarckian Evolution (**EDTL**). Both variants evolve variable-length DTs and perform a simultaneous optimization of the predictive performance (measured in terms of AUC) and model complexity (measured in terms of GE tree nodes). To handle big data, the GE methods include a **training sampling** and **parallelism evaluation mechanism**. Both variants both use PonyGE2 as GE engine, while EDTL uses sklearn DT during the Lamarckian Evolution.

Solutions are represented as a **numpy.where** expression in the format bellow, with *x* being a pandas dataframe with data; *idx* a column from the dataset; *comparison* a comparison signal (e.g., <, ==); *value* being a numerical value; and *result* can be another **numpy.where** expression (creating chained expressions), or a class probability (numeric value from 0 to 1). Due to this representation, current evoltree implementation only allows numerical attributes. More details about this work can be found at: https://doi.org/10.1016/j.eswa.2020.114287.

```
numpy.where(x[<idx>]<comparison><value>,<result>,<result>)
```



```
numpy.where(x['partner'] > 2.32,
    numpy.where(x['app'] < 34, (0.95), (0.6)),
    numpy.where(x['manager'] <= 1,
        numpy.where(x['operator'] <= 2, (0.5), (0.3)),
        (1)
)
```

Figure 25: User Manual: GE-based Decision Tree example.

a.2 Installing

Using pip:

```
pip install evoltree
```

a.3 Quick Start

This short tutorial contains a set of steps that will help you getting started with **evoltree**.

a.3.1 *Load Example Data*

evoltree package includes two example datasets for testing purposes only. Data is ordered in time and the second dataset contains events collected after the first one. To load the datasets, already divided into train, validation and test sets, two functions were created: - **load_offline_data** - returns the training, validation and test sets from first dataset, used for static environments; - **load_online_data** - returns the training, validation and test sets from both datasets, used for online learning scenarios.

Next steps present how to load data in the two different modes (online and offline). Due to privacy issues, all data is anonymized.

```
# Import package
from evoltree import evoltree
# Create evoltree object
edt = evoltree()
# Loading first example dataset, already divided into train, validation and test sets.
X, y, X_val, y_val, X_ts, y_ts = edt.load_offline_data()
# Loading both example datasets, already divided into train, validation and test sets.
X1, y1, X_val1, y_val1, X_ts1, y_ts1, X2, y2, X_val2, y_val2, X_ts2, y_ts2 = edt.
    load_online_data()
print(X)
print(y)

# Outputs
## X (train)
        col1      col2      col3      col4      col5      col6      col7      col8
                 col9      col10
0       0.755951  4.653432  2.767041  0.739897  0.101081  2.401580  2.890712  3.157321
      5.554321  0.865465
1       5.624979  4.782823  0.416159  0.823179  0.101081  2.446735  0.739841  2.564807
      4.552277  0.991334
```

```
2       0.755951  5.365407  3.081596  0.739897  0.101081  3.174113  1.986985  3.193263
      3.378832  0.865465
3       4.436114  6.393779  0.416159  0.823179  0.101081  2.238128  2.066007  2.564807
      3.436435  0.865465
4       7.071106  6.069200  0.416159  0.739897  0.101081  5.100103  0.739841  3.551982
      4.886248  0.991334
...          ...       ...       ...       ...       ...       ...       ...       ...
           ...       ...
708937  0.755951  4.804125  0.416159  0.823179  0.101081  2.807017  2.066007  2.564807
      5.802875  0.865465
708938  0.755951  2.558737  0.416159  0.823179  0.101081  2.558737  2.066007  2.564807
      5.802875  2.173501
708939  0.755951  2.875355  0.416159  0.823179  0.101081  2.572643  0.739841  2.455024
      2.947305  0.991334
708940  0.755951  4.694221  0.416159  0.823179  0.101081  3.568007  0.739841  2.455024
      6.279122  0.991334
708941  6.631839  2.553551  0.416159  0.823179  0.101081  2.446735  0.739841  2.564807
      5.266229  0.991334

[708942 rows x 10 columns]

[708942 rows x 10 columns]

## y (train)
0         NoSale
1         NoSale
2         NoSale
3         NoSale
4         NoSale
           ...
708937    NoSale
708938    NoSale
708939    NoSale
708940    NoSale
708941    NoSale
Name: target, Length: 708942, dtype: object
```

a.3.2 *Offline Learning: Fit EDT and EDTL models*

Next steps present the basic usage of both variants (EDT and EDTL) for modeling the previously loaded data in an offline environment. Furthermore, since all solutions are stored, it is possible to continue the learning process if needed, by using the **refit** function, also presented below.

```
# Imports
from evoltree import evoltree
from sklearn import metrics
import matplotlib.pyplot as plt
# Create two evoltree objects, one for each variant
edt = evoltree()
edtl = evoltree()
# Load dataset
X, y, X_val, y_val, X_ts, y_ts = edt.load_offline_data()
# Fit both versions on train data
## Normal variant:
edt.fit(X, y, "Sale", X_val, y_val, pop=100, gen=10, lamarck=False, experiment_name="
    test")
## Lamarckian variant, doesn't need as much iterations (gen)
edtl.fit(X, y, "Sale", X_val, y_val, pop=100, gen=5, lamarck=True, experiment_name="
    testLamarck")
# Continue Fitting both versions on the same datasets for extra 2 iterations
## Normal variant:
edt.refit(gen=2)
## Lamarckian variant, doesn't need as much iterations (gen)
edtl.refit(gen=2)
# Predict on test data, using the solution with better predictive performance on
    validation data
y_pred1 = edt.predict(X_ts, mode="best")
y_predL1 = edtl.predict(X_ts, mode="best")
# Compute AUC on test data
fpr1, tpr1, th1 = metrics.roc_curve(y_ts, y_pred1, pos_label='Sale')
fprL1, tprL1, thL1 = metrics.roc_curve(y_ts, y_predL1, pos_label='Sale')
auc1 = metrics.auc(fpr1, tpr1)
aucL1 = metrics.auc(fprL1, tprL1)
# Plot results
fig, ax = plt.subplots(1,1, figsize=(5.5,5))
plt.plot(fprL1, tprL1, color='royalblue', ls="--", lw=2, label="EDTL={}%".format(round
    (aucL1, 2)))
plt.plot(fpr1, tpr1, color='darkorange', ls="-", lw=2, label="EDT={}%".format(round(
    auc1, 2)))
plt.plot([0,1], [0,1], color="black", ls='--', label="baseline=50%")
plt.legend(loc=4)
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.savefig("results.png")
```

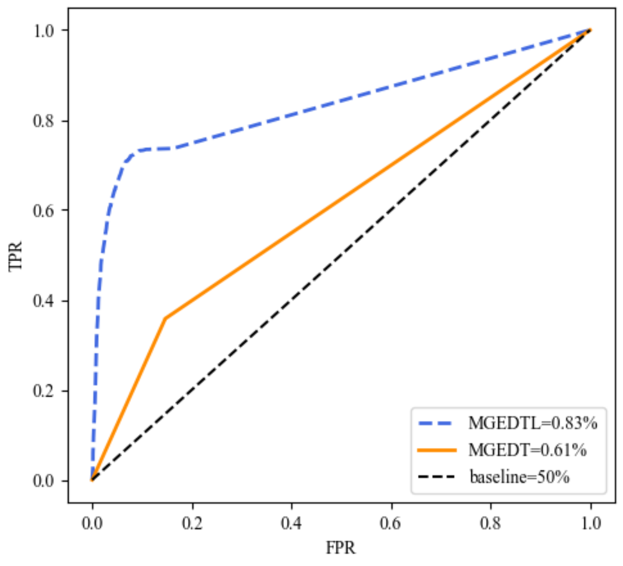


Figure 26: User Manual: ROC curve for EDTs in offline learning task.

a.3.3 *Online Learning: Fit EDT and EDTL models*

EDT variants can both be applied to online learning environments, saving previous solutions and using it as starting point for the learning process, thus, needing a smaller number of iterations to achieve good results. Next steps show how to implement it.

```
# Imports
from evoltree import evoltree
from sklearn import metrics
import matplotlib.pyplot as plt
# Create two evoltree objects, one for each variant
edt = evoltree()
edtl = evoltree()
# Load datasets
X1, y1, X_val1, y_val1, X_ts1, y_ts1, X2, y2, X_val2, y_val2, X_ts2, y_ts2 = edt.
    load_online_data()
### Train models: first dataset
# Fit both versions on train data
## Normal variant:
```

```
edt.fit(X1, y1, "Sale", X_val1, y_val1, pop=100, gen=10, lamarck=False,
    experiment_name="test")
## Lamarckian variant, doesn't need as much iterations (gen)
edtl.fit(X1, y1, "Sale", X_val1, y_val1, pop=100, gen=5, lamarck=True, experiment_name
    ="testLamarck")
# Predict on test data, using the solution with better predictive performance on
    validation data
y_pred1 = edt.predict(X_ts1, mode="best")
y_predL1 = edtl.predict(X_ts1, mode="best")
# Compute AUC on test data
fpr1, tpr1, th1 = metrics.roc_curve(y_ts1, y_pred1, pos_label='Sale')
fprL1, tprL1, thL1 = metrics.roc_curve(y_ts1, y_predL1, pos_label='Sale')
auc1 = metrics.auc(fpr1, tpr1)
aucL1 = metrics.auc(fprL1, tprL1)
### Re-Train models: second dataset
# Fit both versions on train data
## Normal variant:
edt.fit_new_data(X2, y2, X_val2, y_val2, pop=100, gen=5, lamarck=False)
## Lamarckian variant, doesn't need as much iterations (gen)
edtl.fit_new_data(X2, y2, X_val2, y_val2, pop=100, gen=2, lamarck=True)
# Predict on test data, using the solution with better predictive performance on
    validation data
y_pred2 = edt.predict(X_ts2, mode="best")
y_predL2 = edtl.predict(X_ts2, mode="best")
# Compute AUC on test data
fpr2, tpr2, th2 = metrics.roc_curve(y_ts2, y_pred2, pos_label='Sale')
fprL2, tprL2, thL2 = metrics.roc_curve(y_ts2, y_predL2, pos_label='Sale')
auc2 = metrics.auc(fpr2, tpr2)
aucL2 = metrics.auc(fprL2, tprL2)
# Plot results
fig, ax = plt.subplots(1,1, figsize=(5.5,5))
plt.plot(fprL1, tprL1, color='royalblue', ls="--", lw=2, label="EDTL (1)={}%".format(
    round(aucL1, 2)))
plt.plot(fpr1, tpr1, color='darkorange', ls="-", lw=2, label="EDT (1)={}%".format(
    round(auc1, 2)))
plt.plot(fprL2, tprL2, color='navy', ls="--", lw=2, label="EDTL (2)={}%".format(round(
    aucL2, 2)))
plt.plot(fpr2, tpr2, color='tan', ls="-", lw=2, label="EDT (2)={}%".format(round(auc2,
     2)))
plt.plot([0,1], [0,1], color="black", ls='--', label="baseline=50%")
plt.legend(loc=4)
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.savefig("results_online.png")
```

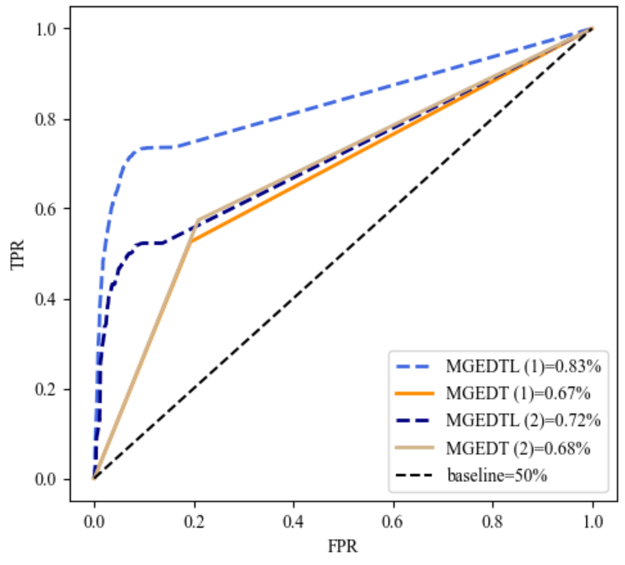


Figure 27: User Manual: ROC curve for EDTs in online learning task.

a.4 Citation

If you use **evoltree** for your research, please cite the following paper:

- Pedro J. Pereira, Paulo Cortez, Rui Mendes. **Multi-objective Grammatical Evolution of Decision Trees for Mobile Marketing User Conversion Prediction.** Expert Systems with Applications, Elsevier, 168:114287, April, 2021

```
@article{DBLP:journals/eswa/PereiraCM21,
    author    = {Pedro Jos{\'{e}} Pereira and Paulo Cortez and Rui Mendes},
    title     = {Multi-objective Grammatical Evolution of Decision Trees for Mobile
        Marketing user conversion prediction},
    journal   = {Expert Systems with Applications},
    volume    = {168},
    pages     = {114287},
    month     = {April},
    year      = {2021},
    url       = {https://doi.org/10.1016/j.eswa.2020.114287}
}
```