Universidade do Minho
Escola de Engenharia

Luís Miguel da Rocha de Matos

An intelligent decision support system for mobile performance marketing

December, 2021

Luís Miguel da Rocha de Matos **An intelligent decision support system for mobile performance marketing**

UMinho | 2021

Universidade do Minho
Escola de Engenharia

Luís Miguel da Rocha de Matos

An intelligent decision support system for mobile performance marketing

PhD Thesis
PhD Thesis in Information Systems

Thesis Supervised by
Professor Paulo Alexandre Ribeiro Cortez
Professor Rui Manuel Ribeiro de Castro Mendes

December, 2021

Universidade do Minho
Escola de Engenharia

Luís Miguel da Rocha de Matos

An intelligent decision support system for mobile performance marketing

PhD Thesis
PhD Thesis in Information Systems

Thesis Supervised by:
Professor Paulo Alexandre Ribeiro Cortez
Professor Rui Manuel Ribeiro de Castro Mendes

December, 2021

ACKNOWLEDGEMENTS

This PhD thesis was a long and arduous journey, composed by many challenges attempts to innovate and solve diverse research issues. Resilience and commitment were essential to make this thesis possible. Many people were critical to transform this PhD work into a success.

My supervisor, Prof. Paulo Cortez was fundamental since the very beginning. His expertise, discipline, rigor and dedication were essential to guide this thesis into the right direction. During this supervision, my willingness to learn and test new challenges was always nourished by the professor. Moreover, his patience, confidence and insight was an incessant encouragement to keep me motivated and push me forward.

I would also like to thank supervisor Prof. Rui Mendes, that provided his intelligence and dedication throughout this thesis with the utmost excellence. Prof. Rui Mendes was perfectly suited to help this project thanks to his competence and expertise in computer science. Thanks to his knowledge, it was possible to create an Intelligent Decision Support System and deploy it into a real environment.

The knowledge shared by my friends during the course of this PhD was important to me to acquire the motivation need and new skills required to execute and finish this project. The Minho University, the ALGORITMI Center and School of Engineering provided the computational resources to implement the experiments and models created.

The cooperation with the OLAmobile company was also very important. OLAmobile kindly provided us with data, defined several of the business rules and provided us useful feedback.

The diverse anonymous reviewers of our publications contributed with helpful insights and suggestions that enhanced the quality of this thesis.

My family was important to my dedication to the PhD studies and I am fortunate to have a devoted family that provides me with constant support and dedication. Finally, my beloved was my anchor. Her insight, dedication, calmness, love and strength was essential to my growth and development of this work. I dedicate this thesis to my beloved family and girlfriend.

STATEMENT OF INTEGRITY

ABSTRACT

Mobile devices (e.g., smartphones, tablets) are becoming commonplace and thus, mobile performance marketing is nowadays witnessing a considerable growth. In this type of marketing, real-time bidding is typically implemented by using a **Demand-Side Platform (DSP)** that matches users to ads. If there is a product purchase (conversion), the DSP facilitates a monetization exchange by returning a percentage of the sale revenue to the publishers. Under this context, a critical DSP issue is the prediction of the user **Conversion Rate (CVR)**, often modeled as a classification task and where the goal is to estimate if there will be a purchase when a user clicks on a dynamic link and then views an ad.

The main objective of this study is the development of an Intelligent Decision Support System (IDSS), which can be integrated into an existing DSP. The IDSS is particularly focused on predicting the CVR, which can be used as a way to select the best advertisements for users (e.g., with the highest conversion probability). In this work, the IDSS was built using *big data* which in turn is generated from a DSP that operates at a global level. In a first phase, a *stream processing engine* was developed in order to obtain data samples from a complete set of events generated by DSP. This sample data included *redirects* and sales events collected between 2017 and 2019. Next, several *Machine Learning (ML)* methods, including *Deep Learning (DL)*, balancing and pre-processing methods were explored. The experimental results demonstrate that the proposed ML approach, which includes a pre-processing of *Percentage Categorical Pruned (PCP)* and a reuse learning of the Deep Feedfoward Network (DFFN) model, obtained the best predictions of CVR (for both binary and multi-class classification), requiring a computational effort that is manageable by DSP. Additionally, the IDSS developed, based on the proposed ML approach, proved to provide a valuable potential impact on the analyzed *mobile marketing* domain.

Keywords: Big Data; Categorical Transformation; Classification; Conversion Rate (CVR); Deep Multilayer Perceptron; Intelligent Decision Support System (IDSS); Ordinal Classification.

RESUMO

Os dispositivos móveis (e.g., *smartphone*, *tablet*, *smart TV*) têm vindo a tornar-se cada vez mais comuns, fazendo com que o mercado *mobile* testemunhe um crescimento enorme nos dias de hoje. Neste tipo de marketing, o leilão é tipicamente implementado usando um ***Demand-Side Platform (DSP)*** que interliga os utilizadores aos anúncios. Se existir a compra de um produto (conversão), o DSP facilita o fluxo de dinheiro, retornando assim uma percentagem da venda aos *publishers*. Sobre este conceito, o factor chave do DSP é a previsão do **Conversion Rate (CVR)**, que é geralmente abordado como uma tarefa de classificação cujo o objectivo é estimar se irá ocorrer uma compra quando o utilizador (*user*) clique num *dynamic link* e veja a publicidade.

O principal objectivo deste estudo é o desenvolvimento de um Sistema de Apoio à Decisão Inteligente (IDSS em inglês), que possa ser integrado num DSP já existente. O IDSS é particularmente focado em prever o CVR, que pode ser utilizado como forma de selecionar as melhores publicidades aos utilizadores (e.g., com a maior probabilidade de conversão). Neste trabalho, o IDSS foi construído utilizando *big data* que por sua vez for gerada de um DSP que opera a nível global. Numa primeira fase, foi desenvolvido um *stream processing engine* com o objetivo de obter amostras de dados de um conjunto completo de eventos gerado pelo o DSP. Esta amostra de dados incluiu eventos de *redirects* e de vendas coletadas entre os 2017 e 2019. De seguida, vários métodos de *Machine Learning (ML)*, incluindo *Deep Learning (DL)*, balanceamento e métodos de pré-processamentos foram explorados. Os resultados experimentais demonstram que a abordagem de ML proposta, que inclui um pré-processamento de *Percentage Categorical Pruned (PCP)* e uma aprendizagem reuse do modelo Deep Feedfoward Network (DFFN), obteve as melhores previsões de CVR (tanto para classificação binária como multi-classe), exigindo um esforço computacional comportável pelo DSP. Adicionalmente, o IDSS desenvolvido, baseado na abordagem de ML proposta, demonstrou fornecer um impacto potencial valioso no domínio do *mobile marketing* analisado.

Keywords: *Big Data*; *Categorical Transformation*; *Classification*; *Conversion Rate (CVR)*; *Deep Multilayer Perceptron*; *Intelligent Decision Support System (IDSS)*; *Ordinal Classification*.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

A	
AUC	Area Under Curve.
B	
BA	Business Analytics.
BI	Business Intelligence.
C	
CNN	Convolutional Neural Networks.
CRISP-DM	The Cross Industry Standard Process for Data Mining.
CTR	Click Through Rate.
CVR	Conversion Rate.
D	
DFFN	Deep Feedfoward Network.
DL	Deep Learning.
DM	Data Mining.
DS	Decision Stump.
DSP	Demand-Side Platform.
DSR	Design Science Research.
DSS	Decision Support System.
F	
FIFO	First-In-First-Out.
G	

GAN Generative Adversarial Network.

I

IAAS Infrastructure-as-a-service.

IDEAL International Conference on Intelligent Data Engineering and Automated Learning.

IDF Inverse Document Frequency.

IDSS Intelligent Decision Support System.

IJCNN International Joint Conference on Neural Networks.

INTS Intelligent Systems.

IS Information Systems.

IT Information Technology.

J

JSON JavaScript Object Notation.

K

KBS Knowledge based Systems.

L

LR Logistic Regression model.

M

MAEO Mean Absolute Error for Ordinal Classification.

ML Machine Learning.

MLP Multilayer Perceptrons.

N

NB Naive Bayes model.

NN Neural Networks.

NOSQL Not Only SQL.

O

OA	Ordinal Approach.
OB	Ozaboost.

P

PA	Predictive Analytics.
PAAS	Platform-as-a-service.
PCP	Percentage Categorical Pruned.
PPC	Pay-Per-Click.

R

RDBMS	Relational Database Management Systems.
RF	Random Forests model.
RH	Random Hoeffding Tree.
RNN	Recurrent Neural Network.
ROC	Receiver Operating Characteristic.
RTB	Real-Time-Bidding.

S

SAAS	Software-as-a-service.
SGD	Stochastic Gradient Descent.
SQL	Structured Query Language.
SSP	Supply-Side Platforms.

T

TSS	True Saved Space.

X

XG	XGBoost model.

1 INTRODUCTION

This chapter presents the motivation and goals of this PhD work, the adopted research methodology, bibliographic search strategy, the main contributions and how this document is structured.

1.1 Motivation

Due to the massive usage of portable computing devices (e.g., smartphones, tablets), there has been a worldwide increase in the domain of mobile marketing. With the rise of such powerful and effective ways of digital marketing, there is a need for more powerful and efficient ways of extracting and using the knowledge contained in this type of raw data with the goal of better reaching the intended audience (*Teixeira et al.*, 2018). In effect, developments in **Information Technology (IT)** have led to improved approaches and tools to store and process data (e.g., Big Data, Machine Learning, Predictive Analytics), which can potentially provide gains in the field of digital marketing.

In mobile performance marketing, monetary compensation only occurs when an ad performs well (e.g., a product purchase). Real-time bidding is implemented using a technological system called a **Demand-Side Platform (DSP)**. Both publishers and advertisers use a DSP. Publishers are web content owners (e.g., online games, news portals) that attract a vast audience of users. The web content is funded by using dynamic link ads provided by the DSP. To access the web content, users need to previously click the dynamic link ad. When such an ad is clicked, the DSP redirects the user to a specific marketing campaign from an advertiser. If there is a product purchase (known as a conversion), the DSP facilitates the cash flow, returning a percentage of the sale revenue to the publishers.

A DSP typically generates big data due to its worldwide usage and high velocity in which the ad clicks are generated. Under this context, there are two key prediction goals, which are often modeled as binary classification goals:

- **Click Through Rate (CTR)** – to detect if a user will click an ad when browsing a web page or using an app; and

- **Conversion Rate (CVR)** – where the goal is to estimate if there will be a purchase when a user clicks on a dynamic link and then views an ad.

1.2 Goals

This PhD work aims to research the design of an **Intelligent Decision Support System (IDSS)** that focuses on mobile user **Conversion Rate (CVR)** prediction. This PhD was inserted within the R&D project "PROMOS - PRediction and Optimization of MObile Subscription marketing campaigns", which involved a worldwide mobile marketing company, OLAmobile[1]. Using recent data from the OLAmobile Demand-Side Platform (DSP), this PhD aims to design an IDSS to support the DSP user to ad matching decisions. Therefore, the PhD research question can be formulated as: **"How to design an Intelligent Decision Support System (IDSS) that uses mobile performance marketing data events and that is capable of providing value to a DSP?"**

The main PhD thesis goal was reached by addressing the several intermediate objectives, namely:

- Investigating and developing on a new automatic algorithm for the prediction and optimization of advertising campaigns. In particular, the algorithm should select the best mobile product to show the end user.
- Creating and deploying a stream engine collecting system. The analyzed DSP stored all events in a data center. Due to technological restrictions (e.g., communication and storage costs), it was not possible to work, at an initial stage, with the full DSP big data. Thus, this issue was addressed by designing a sampling stream engine. The engine must be capable of fetching, treating and storing data samples based on several key aspects (e.g., size of file, number of days we want to collect, automatically adapt the number of fetching records since there is an delay between the redirect record and the processing of the sale record).
- Explore, adjust, compare and evaluate several Machine Learning (ML) methods that can be applied to the user Conversion Rate (CVR) prediction (including both binary and ordinal classification). This exploration involved several computational experiments, which included distinct preprocessing methods and balancing training methods.
- Execute a rigorous evaluation of the ML models, preprocessing methods and balancing methods when using the DSP data. This evaluation consists of several statistical tests, the use of several quality assessment measures, such as Area Under Curve (AUC), F1-Score, Mean Absolute Error for Ordinal Classification (MAEO) and computational effort measures regarding rolling holdout window, preprocessing of data, (re)training, updates to the model and predictions.
- Create and deploy an IDSS campaign simulator that is aligned with the company objectives. This simulator will consider the data-driven prediction models' insights to assign ads to users. The campaign simulator was also evaluated using diversity measures based on the information entropy concept.

1 https://www.olamobile.com/

1.3 Bibliographic Search Strategy and Research Methodology

This section focuses on the bibliography search strategy and the research methodology used for this work, namely **Design Science Research (DSR)**.

1.3.1 *Bibliography Search Strategy*

The related research works were obtained by two executing main procedures. Firstly, several generic textbooks and specific studies were provided by the supervisors, allowing an initial reading of the relevant terms and some pertinent works. Secondly, several keyword combinations were explored in scientific databases, such as: **"Marketing"**, **"Mobile Marketing"**, **"Mobile Performance Marketing"**, **"Click Through Rate (CTR)"**, **"Conversion Rate (CVR)"**, **"Prediction"**, **"Machine Learning"**, **"Deep Learning"**. The obtained documents were manually reviewed, in order to check their pertinence for the thesis. Two main scientific database engines were used to search for the relevant works: Scholar Google[2] and Science Direct[3]. Overall, the queries resulted in hundreds of documents that were reviewed and filtered manually, checking first the title, then the abstract and finally the whole document. Recently (e.g., published in the last 10 years) and quality (e.g., published in international journals) factors were also considered when selecting the relevant works. Since this PhD thesis also included the development of technological artifacts, as a complementary technical resource, several online courses were attended, aiming to better understand some specific tools (e.g., `R` and `Python` environments, `MongoDB` database, `keras` module) and computational concepts (e.g., parallelism).

1.3.2 *Design Science Research Methodology (DSR)*

This project focuses on the design and development of a IDSS tool for solving a business need. Thus, this PhD thesis can be naturally supported by the **Design Science Research (DSR)** methodology. The DSR consists of a set of steps to assist in the development of an novel **Information Systems (IS)** artifact. When applied to the design of a IDSS that assumes data-driven models, based on **Data Mining (DM)** and **Machine Learning (ML)** approaches, the adopted DSR methodology shares several similarities with two other methodologies: the **four stage model proposed by** *Simon* (1960) and *Sprague Jr* (1980); and the **The Cross Industry Standard Process for Data Mining (CRISP-DM)** (*Chapman et al.*, 2000) methodology (detailed in Section 2.6). Fig. 1 shows the similarities between the three methodologies, in an adaptation of (*Moro et al.*, 2015b). While similar, it should be noted that there are some differences between the three methodologies. DSR is a more generic

2 `https://scholar.google.pt`
3 `https://www.sciencedirect.com`

research methodology, widely used in the Information Systems (IS) discipline and that can support the design of Decision Support System (DSS) (*Arnott and Pervan*, 2014). The Simon's decision making model was the first known methodology to support the design of DSS (*Simon*, 1960). Later, the model was complemented by the work of *Sprague Jr* (1980), resulting in a four stage decision making model. Finally, CRISP-DM is a methodology proposed by *Chapman et al.* (2000) to increase the success of DM/ML projects, typically involving both ML and business experts (from the targeted organization or company). More details about the Simon's decision making model and CRISP-DM methodology are presented in Sections 2.5 and 2.6.

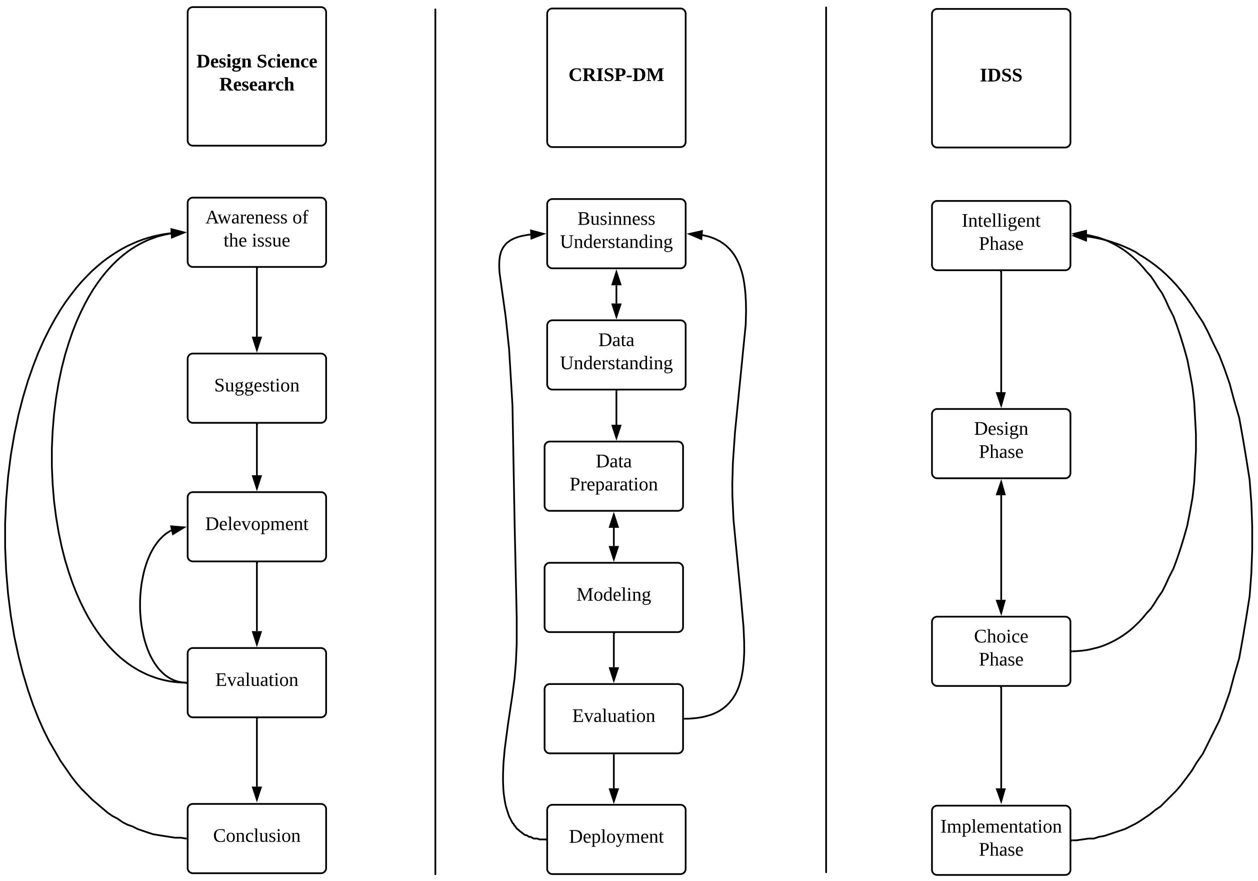

Figure 1: Design Science Research (DSR) aligned with *CRISP-DM* and *Simon's Decision Making Model*, adapted from *Moro et al.* (2015b)

In this PhD work, we used 5 steps of **DSR**, namely:

1. **Awareness of the issue** – As shown in Fig. 1, this step is aligned with the "Business Understanding" stage of CRISP-DM and "Intelligence" phase of the Simon's model. The addressed issue was identified by the DSP company and it is related with the PROMOS R&D project: to design a ML algorithm that could add value to a DSP by assigning ads to users. Mobile user CVR prediction is a complex real-world task due to four main reasons. First, it involves big data, since typically millions of clicks are generated every hour. Second, only a tiny amount of user clicks are converted into a sale. Third, a limited set of data features is available, due to technological constrains and privacy issues (e.g., it is not possible to identify a single user). Fourth, data

features are mostly categorical, often presenting a large cardinality with hundreds or thousands of levels. As detailed in Chapter 2, user CVR prediction has been mostly modeled using linear models (e.g., logistic regression), assuming the one-hot categorical encoding and static offline learning environments. Also, most studies focus only on classification performance but not on computational effort, which is a critical issue, since DSPs require constant model updates and real-time predictions. All these issues make the design and developing of a IDSS within this area a challenging task, which is also a research opportunity.

2. **Suggestion** – Through a sequence of several research works (see Section 1.4 and Chapter 3), we have designed, adjusted and empirically compared several ML approaches for mobile user CVR prediction. These approaches consider distinct preprocessing methods, namely distinct training balancing and nominal to numeric transforms and ML algorithms (e.g., based on off-line and online training methods). The final solution assumes a newly proposed PCP transform (to handle high cardinality attributes) and a novel *reuse* **Deep Feedfoward Network (DFFN)**, which can model the binary task (as a probability) of mobile user CVR estimation. This model was included IDSS simulator, aiming to realistically measure its value for user ad matching.

3. **Development** – This step focuses on the development of the artifact itself. This development was made incrementally and iteratively, throughout the sequence of the several papers and outcomes that were produced. We used several technologies: the `R` (*R Core Team*, 2016) and `Python` (*Rossum*, 1995) programming languages and several of their packages/modules (e.g., `rminer`, `keras`). To obtain and process the DSP data, we also developed a stream processing engine and used fast storage technologies, such as `MongoDB` (`https://www.mongodb.com/`).

4. **Evaluation** – This step consists in validating the artifact and planning possible improvements. During this phase, we used a realistic evaluation procedure and diverse predictive performance and computational effort measures, namely:
   - **Rolling Window** procedure, which simulates a classifier usage through time, with multiple training and test updates (*Tashman*, 2000). This procedure was adopted in all ML mobile user CVR prediction experiments.
   - **Area Under Curve (AUC)** of the **Receiver Operating Characteristic (ROC)** curve, which measures the degree of discrimination of a binary classifier (*Moro et al.*, 2015b; *Du et al.*, 2016). The AUC was adopted as the predictive performance measure for all binary user CVR prediction modeling experiments.
   - **F1-Score** metric, which considers both Precision and Recall scores (*Witten et al.*, 2017). The global measure is obtained by using the Macro-averaging F1-score (MF1), which weights equally the F1-score for each class. This metric was adopted in the multi-class (ordinal) ML experiments.

- **Mean Absolute Error for Ordinal Classification (MAEO)** metric, which computes how far (using absolute errors) are the predictions from the target (*Sousa et al.*, 2013). The MAEO metric was also used in the mult-class experiments.
- **Time elapsed** for preprocessing, training and inference (in order to measure the computational effort).
- For the final IDSS realistic simulation experiments we also measured the **potential profit**, which corresponds to the expected increase of sales; and **advertisers diversity**, evaluated using two measures, the **campaign Variety** and a proposed **True Saved Space indicator**, inspired by the Entropy concept proposed by (*Shannon*, 1948).

5. **Conclusion**: This final step signals the end of the creation of the artifact, when the robustness of the system is considered satisfactory. In this thesis, with corresponds to the tested IDSS simulator.

1.4 Contributions

The main contributions of this PhD dissertation are:

- A comparison study involving several ML approaches (with different preprocessing and ML algorithms) that were adjusted for mobile user Conversion Rate (CVR) prediction (*Matos et al.*, 2018). To perform this study, we developed first a stream engine for mobile marketing data collection. Afterwards, we compared several ML models for different data sampling periods (e.g., 30 minutes, 1 day, 1 week), traffic modes (BEST and TEST), and sampling methods (collected and realistic). This initial exploratory study provided initial insights about which preprocessing and ML methods tend to work better with the analyzed DSP mobile marketing data.
- A novel **Deep Learning (DL)** approach that considers a newly proposed **Percentage Categorical Pruned (PCP)** transform and a **reuse Deep Feedfoward Network (DFFN)** (*Matos et al.*, 2019a). To collect the data, we further enhanced the stream engine, retrieving more recent data from the DSP data center. The PCP transformation is a much more lightweight numberic encoding of categorical data and that only maps the frequent levels of a given attribute. It should be noted that the popular one-hot encoding can generate thousands of features (1 for each unique categorical value), which heavily increases the computational memory and effort to handle such attributes. Moreover, we also compare two DFFN learning modes, offline (reset) and online (reuse). The reset mode is the standard offline learning method where a new DFFN model is fully initialized with random weights when new training data is available. In contrast, the reuse approach assumes an online learning approach, where the weights of the previously trained DFFN are first stored and then “reused”.

- An exploratory study on ordinal classification of mobile marketing user CVR (*Matos et al.*, 2019b). When compared with previous binary modeling, the ordinal classification is potentially more valuable for the analyzed DSP, since each class is assigned with a conversion revenue level. Several ML experiments were held. The best results were achieved by the proposed reuse DFFN. However, after obtained feedback from the DSP company, it was clear that the obtained predictions were far from the ideal outcome, thus the ordinal model was discarded from the IDSS implementation.

- An Intelligent Decision Support System (IDSS) that is capable of selecting mobile marketing campaigns for users according to two different strategies and that includes the best model proposed in (*Matos et al.*, 2019a). A realistic offline DSP simulation was executed, aiming to validate the two ad assignment strategies: best campaign selection and a random selection within the top best candidate ad campaigns. The two IDSS strategies were compared with the ad matching procedure currently adopted by the analyzed DSP in terms of three criteria: response time; potential profit, in terms of expected increase of sales; and advertisers diversity, measured in terms of campaign Variety and a proposed True Saved Space indicator.

Associated with these contributions, the work of this PhD dissertation resulted in four scientific publications and one technological output:

- Conference Papers:
  - Matos, L. M., P. Cortez, R. Mendes, and A. Moreau, 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*, pp. 140–146, 2018
  - Matos, L. M., P. Cortez, R. Mendes, and A. Moreau, Using deep learning for mobile marketing user conversion prediction, in *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*, pp. 1–8, IEEE, 2019a
  - Matos, L. M., P. Cortez, R. C. Mendes, and A. Moreau, 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*, *Lecture Notes in Computer Science*, vol. 11871, pp. 60–67, Springer, 2019b

- A journal article that was submitted:
  - Matos, L. M., P. Cortez, R. Mendes, and A. Moreau, A deep learning based decision support system for mobile performance marketing, *Submitted to a journal.*

- Development of a Python Module[4]:
  - Matos, L. M., P. Cortez, and R. C. Mendes, Cane - categorical attribute transformation environment, 2020

1.5 Thesis Organization

Excluding this first Chapter, this dissertation is organized as follows:

- **Chapter 2** introduces some general background concepts related to this PhD project, namely: digital and mobile marketing; Big Data; cloud and distributed computing; IDSS; Data Science, Data Mining and Predictive Analytics; ML and Deep Learning; preprocessing and balancing methods; programming languages for data science; and evaluation measures. Then, more specific details are presented about the "core" state-of-the-art studies, involving ML approaches for user CTR and CVR prediction.

- **Chapter 3** describes the methods, experiments and results produced within this PhD work and that led to three conference publications and one article submitted to a journal. Section 3.1 presents the initial set of experiments, related with an exploratory study of user CVR (binary classification task) by several ML approaches. Section 3.2 represents the second set of experiments, which were focused on DL models, such as Deep Feedfoward Network (DFFN) and Convolutional Neural Networks (CNN) models. Section 3.3 explores an ordinal classification for user CVR prediction. Finally, Section 3.4 addresses a realistic IDSS simulation, in which the best binary DL user CVR prediction model is used to assign ads to users, under two main strategies.

- **Chapter 4** summarizes the main conclusions of this PhD thesis and discusses the achieved results. Also, it suggests future research work in this area.

4 $38,525$ downloads according to the website: `https://pepy.tech/project/cane` since its release in June 3^{rd} of 2020.

2 BACKGROUND

This chapter introduces the general background concepts that are related with this PhD project. Then, it details the state-of-the-art studies that employed data analytics for predicting user CVR in mobile marketing performance systems.

2.1 Digital and Mobile Marketing

In the digital era the internet has been playing an important role in conveying and opening new forms of marketing. As such, the term Digital Marketing was coined in 1990s as the digital era took advantage from the Web 1.0 platform. These first platforms allowed the users to assess and use the information they needed albeit it was not possible to share it on other platforms. Then, with the surge of the Web 2.0 people became more active and allowed the interaction between businesses and end-users. More sophisticated Web places, such as social media and search engines, started to emerge and opened more opportunities for businesses to display their products/services to the general public and with the advantage of being available all the time. An important milestone in this field was the creation of the cookie technology. Advertisers began to look for other ways to capitalize on the this recent technology. One such technique was to track browsing habits and usage patterns of frequent internet users to tailor advertisements adequate to their tastes. The cookies were devised to record user habits, offering marketers several ways to collect user data (*Monnappa*, 2021).

Some key applications for the digital marketing field include (*Barone and James*, 2020):

- Website Marketing – A website is the centerpiece for digital marketing projects. It is a compelling channel, thus being the medium needed to execute several online marketing campaigns. A website should represent a brand, product, and service clearly and memorably. It should be fast, mobile-friendly, and easy to use.

- Pay-Per-Click (PPC) Advertising – PPC advertising enables marketers to reach users on a high number of digital platforms through paid ads. Marketers can set up PPC campaigns on key websites (e.g., *Google*, *Bing*, *LinkedIn*, *Twitter*, *Pinterest*, or *Facebook*) and display their ads based on search terms related to the products or services. PPC campaigns can also segment users based on their demographic characteristics (e.g., age, gender) or target their particular interests or location. Some popular PPC platforms are *Google Ads* and *Facebook Ads*.

- Content Marketing – Content Marketing aims to promote the product or service based on the usage of the content. The content is usually published on a website and then promoted through social media, email marketing or even PPC campaigns. Generally, it is conveyed through blogs, ebooks, podcasts and others.
- Email Marketing – Email marketing is yet one of the most efficient digital marketing channels. There is the misconception of confusing email marketing with spam email messages. Email marketing should be viewed as the medium to get in touch with potential customers or people interested in a specific brand. Many digital marketers use other digital marketing channels, which then leads to their email lists. Consequently, through email marketing, they create customer acquisition funnels to turn those leads into customers.
- Social Media Marketing – The main goal of social media marketing is to increase brand awareness and to establish social trust. This also allows the businesses to get leads or even direct contacts for their products and services.
- Affiliate Marketing – Affiliate marketing uses influencers to promote other people's products, and in return, a commission is earned every time a purchase is made, or a lead is introduced. For instance, some companies like Amazon have affiliate programs and rewards for websites that sell their products.
- Video Marketing – Nowadays, platforms such as youtube and TikTok are being used for buying decisions. Video Marketing allows the user to check and analyze how the product/service works and assess their opinions about the product/service. This type of marketing has been increasingly adopted with the rise of social media platforms.
- SMS Marketing – Companies and nonprofit organizations use SMS to send information about their latest promotions or products to willing customers. In some instances, Political candidates running for office also use SMS message campaigns to spread positive information about their platforms.

With the evolution of IT portable devices such as smartphones facilitate internet services, applications and other services, facilitating the information flow among businesses. These devices became the preferential target for the marketing efforts of most companies, hence the creation of the mobile marketing field. Mobile marketing is related to the usage of marketing campaigns that target mobile devices. It consists of either a two-way or multi-away communication/promotion of an offer between an advertising company and the end consumer. Due to its digital nature, mobile marketing is very interactive, and it is responsible for changing traditional business models into the modern and more relationship based ones (e.g., knowing the user preferences). As such, several key aspects are often considered when interacting with mobile marketing (*Shankar and Balasubramanian*, 2009):

- **Location-specificity** – Many mobile devices, including tablets and mobile phones, have GPS capabilities that identify their geographical position. This allows marketers to choose promotional offers for users based on their location. In contrast with conventional marketing media, billboards also allow location-specific messages and mobile devices to be targeted at the individual user's location based on their stated preferences and revealed behaviors.
- **Portability** – An essential benefit of mobile devices to consumers is their reduced size and ease of carrying. Thus, it is a constant companion to the user, making it easier for marketers to target, communicate and present advertisements based on several factors (e.g., location, last web-searches).
- **Untethered/wireless feature** – Unlike other frequently used devices (e.g., Desktop Computers), the typical mobile device is not tethered or connected by wires for most of its use. This property promotes increased usage, creating more opportunities for marketers to convey marketing messages. At the same time, the short duration of its typical use forces marketers to be concise in their messages. The critical differences between mass marketing (typically conducted through mass media such as magazines and television) and mobile marketing are detailed based on these properties. Mass marketing addresses a broad range of existing and potential customers. Mobile marketing, in contrast, is restricted to owners of mobile devices, and in many cases, to a subset of those owners who opt-in to receive communications from marketers.

This work addresses Mobile Performance Marketing, which typically uses a **Demand-Side Platform (DSP)** to reach consumers in real-time, generating vast amounts of data in the process, which can be used to infer if a user will click on an ad, measured in terms of **Click Through Rate (CTR)**, or the acquisition of a product or service when a user sees an ad, measured in terms of **Conversion Rate (CVR)**.

2.2 Big Data

Big Data refers to databases and other data technologies that can manipulate a substantial volume of data. These databases can reach high volumes of data, such as petabytes. The datasets inside these databases can proliferate due to high-velocity streams (e.g., mobile phone data, software logs, video logs, microphone streaming). The Big Data term is often associated with the 7Vs, as described in the work of *Mikalef et al.* (2017):

- **Volume** – The databases usually harbor a substantial volume of information;
- **Velocity** – Usually obtained in real-time;

- **Variety** – Data is collected through heterogeneous devices, and in the end, the information is aggregated using data-fusion;
- **Veracity** – Concerning the degree of veracity, or accuracy in the data;
- **Value** – The value is how the insight or benefits can be obtained and transmitted to end-users;
- **Variability** – Concerns how insight from media is constantly changing over time and how it can be interpreted in different ways; and
- **Visualization** – It can be described as visually interpreting patterns or trends on the data.

Not Only SQL (NoSQL) is a special type of unstructured database that is commonly used in big data environments because of its speed and scalability when compared with traditional databases, which use the **Structured Query Language (SQL)** programming language. These NoSQL databases were built in the early 2000s to deal with large-scale databases, clustering for cloud and web applications, among other applications. In effect, **Relational Database Management Systems (RDBMS)** became impracticable due to the requirements needed in terms of performance and scalability for big data environments (*Meier and Kaufmann*, 2019). Thus, NoSQL databases are becoming commonplace, being adopted by well-known companies, such as Amazon, Facebook and Google.

There are several computational tools to deal with Big Data structures. A popular tool is MongoDB, that was created by *Merriman et al.* (2007). *MongoDB* stores data in a flexible **JavaScript Object Notation (JSON)** such as text documents, which implies that fields can vary from document to document, and the data structure can be changed over time. MongoDB is an open-source database engine that works with JSON data, allowing us to store data on demand and export it in the same format. This engine also provides a **First-In-First-Out (FIFO)** structure for the tables, known as collections, allowing older data to be overwritten by most recent data. It is used in large companies, such as *Forbes*, *Bosch*, *Comcast*, and others.

Another tool popular Big Data tool is Kafka. Kafka is open-source software, created by *Garg* (2013). It stores messages that come from many arbitrary processes named "producers". Data can be partitioned in different *partitions* within *topics*. In a partition, the messages are indexed and stored together with a time-stamp. Other processes called *consumers* can query messages from partitions. It is widely used by companies, such as *Apple Inc.*, *Betfair*, *Cisco Systems* and others.

2.3 Cloud Computing

Cloud computing delivers several services through the internet. Those resources include tools or applications (e.g., data storage, servers, databases, networking, and software) which, rather than maintaining files on a proprietary hard drive or local storage device, make it possible to save them into a remote database. There are a plethora of cloud computing services, which include:

- Email;
- Storage, backup, and data retrieval;
- Creating and testing apps;
- Analyzing data;
- Audio and video streaming; and
- Delivering software on demand.

Cloud computing is commonly applied as a standard option for several businesses which reduces costs, increases productivity, speed, efficiency, performance, and security. Cloud computing includes three main different service modes (*Frankenfield*, 2020; *AVINetworks*, 2021):

- **Software-as-a-service (SaaS)** – in this model, the system is not marketed as a product but as a service, as its name suggests. Thus, no program is installed on the equipment, and the applications are used over the internet. There are free versions (e.g., *Gmail*) and paid versions (e.g., *Office 365*), which meet personal and professional routines. Therefore, companies can now rely on cloud management systems and a customized solution according to the needs of the business.
- **Infrastructure-as-a-service (IaaS)** – is a cloud computing service where enterprises rent or lease servers for computational and storage services on the cloud. The users can then run any operating system/applications on the rented servers without the need for maintenance or operating costs of those servers. An advantage of this cloud computing business model includes giving customers access to servers in geographical locations close to their end-users. IaaS scale automatically depending on the demand and provide service-level agreement (SLA) regarding the uptime and its performance.
- **Platform-as-a-service (PaaS)** – this is considered the most complex of the three layers of cloud-based computing. Although PaaS shares some similarities with SaaS, the main difference is that instead of delivering online software, it is actually a platform to create software that is delivered via the internet (e.g., *Salesforce, Heroku*).

2.4 Distributed Computing

Distributed computing is where the software system components are shared among several computers/processors. Although the components are spread out across several computers, they are run as one system. This can improve the efficiency and performance of the software.

In the most ubiquitous sense of the term, distributed computing proposes that something is shared among multiple systems that can be located in different areas. Distributed computing may also require some expertise in tooling and soft skills.

A distributed system can consist of many possible configurations, such as mainframes, personal computers, workstations and even minicomputers. This offers many benefits over the centralized systems, including (*IBM*, 2014):

- **Scalability** – The system can easily be expanded by adding more machines as needed.
- **Redundancy** – Some machines can provide the same service(s); if one is unavailable, the work does not stop. Additionally, since many smaller machines can be used, this redundancy does not need to be prohibitively expensive.

Distributed systems use several computers to solve a common problem, where computation is distributed among the connected computers (nodes). Some applications include the work of *Kaufmann* (2011) that detailed how to perform and program in a distributed GPU environment that exceeded the capabilities of a single node GPU; and the *Sterling et al.* (2018) work that implemented the widely known *MapReduce* (*Dean and Ghemawat*, 2008).

2.5 Intelligent Decision Support Systems

A **Decision Support System (DSS)** is an application that analyses business or organizational data and helps the end user to make decisions faster and easier. According to *Rouse* (2010), a **Decision Support System (DSS)** might gather data about several aspects, including:

1. comparative sales between time intervals;
2. projected revenue based on sales assumptions; and
3. several outcomes from different decision alternatives regarding past data or context used.

A DSS is often built using *Simon's decision-making model phases*, which are described as follows (*Campitelli and Gobet*, 2010):

- **Intelligent Phase** – consists of surveying the environment for situations that demand decisions to be taken. This phase is broken down into identifying problems, information gathering, goals and evaluation criteria.
- **Design Phase** – consists in the following step after the intelligent phase, and it is responsible for delineating and analyzing the different courses of action.
- **Choice Phase** – related with the search for decision alternatives and selecting the best solution; and

- **Monitoring Phase** – ensures that the choice that was made provides an increased value to the considered domain.

An **Intelligent Decision Support System (IDSS)** is a DSS that makes use of Artificial Intelligence techniques. Since the 1980s, several Intelligent Decision Support System (IDSS) were proposed based on **Knowledge based Systems (KBS)** and other Intelligent Systems. *Arnott and Pervan* (2014) concluded that the research in **DSS** represents around 10% of the **IS** field, involving nowadays mostly data-driven approaches, namely **Business Intelligence (BI)**. The authors also provide insight into the evolution of DSS, as described in Figure 2. In effect, the latest DSS trend is the usage of **Business Analytics (BA)**, which include **BI** Predictive Analytics and Optimization.

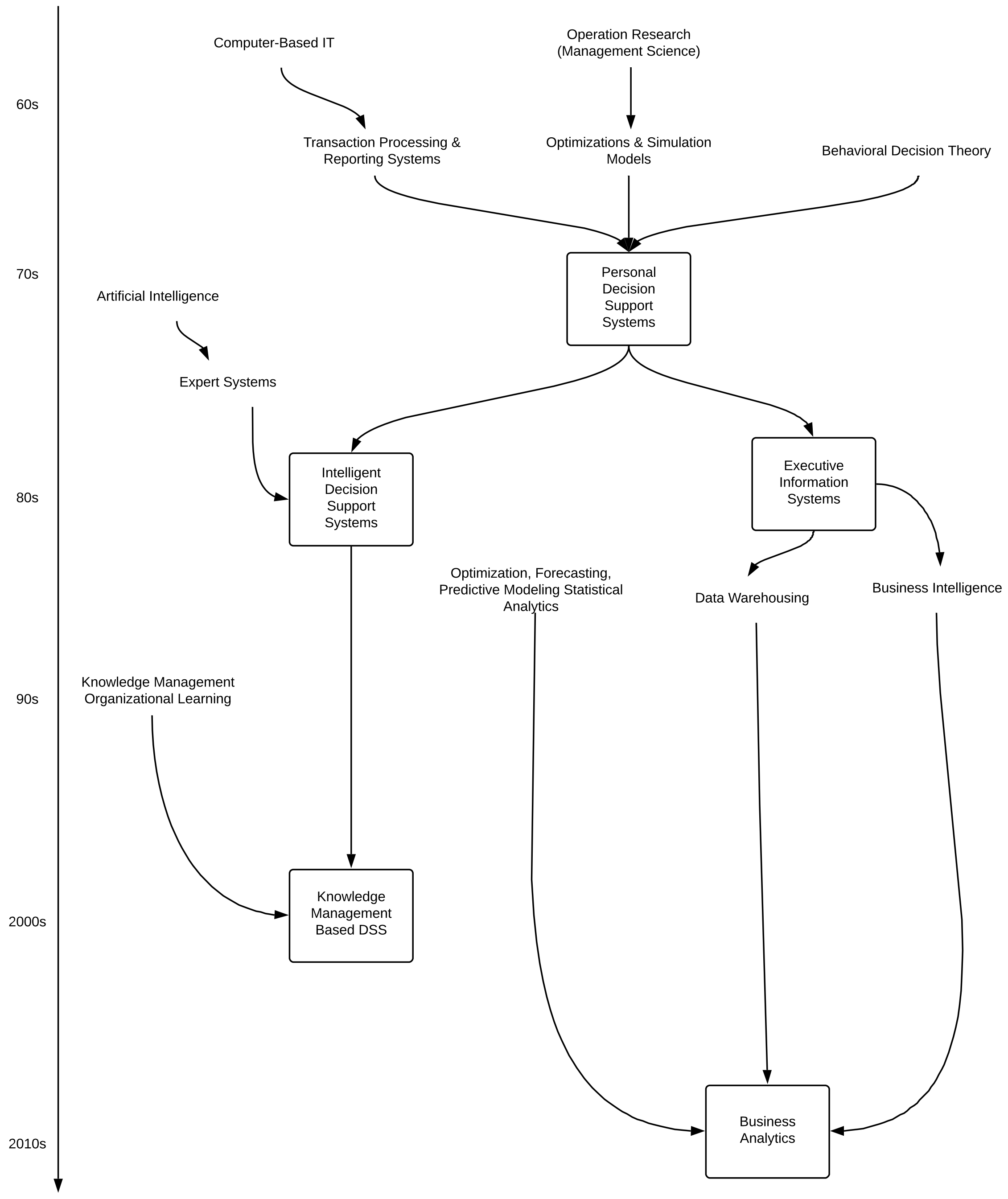
60s
70s
80s
90s
2000s
2010s
Computer-Based IT
Operation Research (Management Science)
Transaction Processing & Reporting Systems
Optimizations & Simulation Models
Behavioral Decision Theory
Personal Decision Support Systems
Artificial Intelligence
Expert Systems
Intelligent Decision Support Systems
Executive Information Systems
Optimization, Forecasting, Predictive Modeling Statistical Analytics
Data Warehousing
Business Intelligence
Knowledge Management Organizational Learning
Knowledge Management Based DSS
Business Analytics

Figure 2: Evolution of a **Intelligent Decision Support System (IDSS)** and **Business Analytics (BA)**, adapted from *Arnott and Pervan* (2014)

2.6 Data Science, Data Mining and Predictive Analytics

The growing interest in collecting data has led to several data-related terms that aim to analyze raw data and extract valuable insights. This includes the terms Data Science, Data Mining and Analytics, which are often viewed as synonyms.

According to *Dhar* (2013) the term "data science" (originally used interchangeably with "datalogy") was used initially as a substitute for computer science by Peter Naur in 1960. However, nowadays, Data Science is the interdisciplinary discipline of scientific methods, processes and systems that extract knowledge or insight on the data, whether it is structured or unstructured. Its primary purpose is to unify all statistics, analytics in order to understand and analyze the phenomena, acting on the fields of mathematics, statistics, information science, and computer science, in particular from the subdomains of Machine Learning, Classification, Cluster Analysis, Data Mining, Databases, and Visualization.

Data Mining has been used to denote the extraction of useful knowledge from raw data, typically by addressing huge amounts of data, in a process that is automated or semi-automated. In this field, the main outcome is a model or a technique that helps to explain patterns in the data and predict future values and is composed by the following steps (*Witten et al.*, 2017):

- **Data Cleaning** – The process in charge of removing noise and cleaning inconsistent data;
- **Data Integration** – Which sources may be combined and discarded;
- **Data Selection** – Selection of important data relevant for the analysis;
- **Data Transformation** – Transformation and consolidation of forms appropriate for mining the data, with aggregating functions or summary operations;
- **Data Mining** – The process in charge of extracting the information from the previous phases;
- **Pattern Evaluation** – Knowledge retrieval and analysis of interesting patterns found; and
- **Knowledge Presentation** – visualization and knowledge representation for end users.

Regarding **Predictive Analytics (PA)**, it utilizes a variety of statistical techniques that range from predictive modeling, Machine Learning and Data Mining that analyzes current and historical facts in order to make predictions about future or unknown events (*Nyce*, 2013; *Eckerson*, 2007). Nowadays, **PA** has received more notoriety, given the growing amount of available data and the power that this type of analytics possesses. This field can be used in almost any domain area, such as finance, healthcare, fraud, insurance, government and employment (*Shmueli and Koppius*, 2011).

Another interesting framework that is widely used is **The Cross Industry Standard Process for Data Mining (CRISP-DM)**. **CRISP-DM** is a popular methodology used to improve the success of Data Mining projects, and it is composed of six main phases (*Chapman et al.*, 2000):

- **Business Understanding**: understanding the project objectives and requirements;

- **Data Understanding**: this phase starts by collecting data and understanding it; this phase also aims to identify quality problems in the data collected;
- **Data Preparation**: involves several data processing steps, such as handing missing data, outlier detection and removal, data and variable selection;
- **Modeling**: In this phase, several Machine Learning algorithms are explored to fit the data-driven models;
- **Evaluation**: analysis of the best data-driven models obtained in the previous phase; in particular, aspects such as novelty, relevance and/or usefulness to the business domain are considered in this phase; and
- **Deployment**: aiming to implement and maintain the Data Mining models for increasing business value.

Figure 3 presents the **CRISP-DM** phases. It is interesting to note that the phases can interact with one and another. Also, the methodology contemplates an iteration between the business and data analysis experts throughout most of its phases.

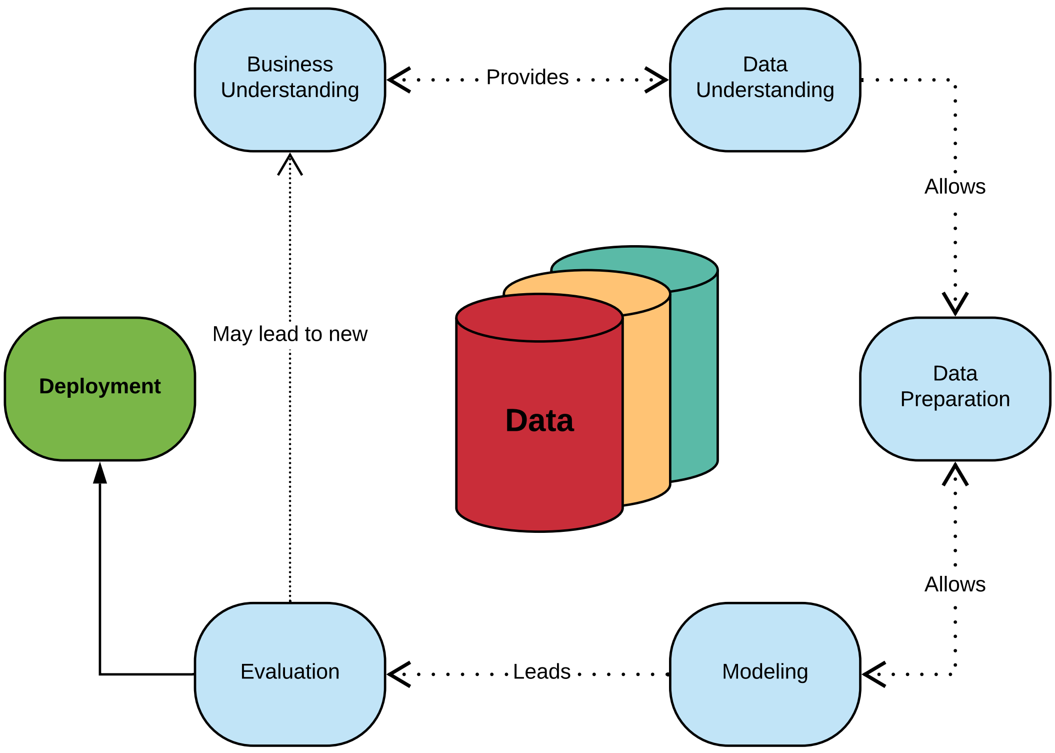

Figure 3: CRISP-DM methodology, adapted from *Chapman et al.* (2000)

2.7 Data Streams

Data streams are algorithmic abstractions that support real-time analytics, usually represented by sequences of items, possibly infinite, with each item containing a timestamp. They often represent a sequence of temporal events. There are two main algorithmic challenges when dealing with streaming data. Firstly, the stream is vast and generated with a high velocity, thus there is a need to process the data in real-time. Approximate solutions that require less computational time and memory are often used to get the most proximate representation of the data. Secondly, the data tends to change through time, resulting in several types of temporal dynamics (e.g., in the features collected, in the concepts to be learned) (*Bifet et al.*, 2018). In particular, when approaching classification tasks there can be changes to the distribution of the target output, which is often termed as **concept drift**. This can produce a degradation over time in the performance of a **Machine Learning (ML)** model. The concept drift effect can be handled by using an Adaptive Learning, which can be implemented with incremental or ensemble learning (*Janardan and Mehta*, 2017).

2.8 Machine Learning

ML investigates how computers can learn from data. The aim is for computer programs to automatically learn to recognize complex patterns and make intelligent decisions based on data (e.g., image recognition, postal code handwriting). **ML** is considered to be a fast-growing discipline (*Witten et al.*, 2017) . In this research project, we are focusing on a specific type of **ML** which is based on supervised learning.

In **ML** it is common to use Offline Learning to train and deploy **ML** models. Offline Learning, also known as batch learning, involves analyzing, learning and testing the data in an offline environment with a static dataset. Examples of commonly used Offline Learning models for classification purposes include (*Du et al.*, 2016; *Zhang et al.*, 2014; *Moro et al.*, 2015b, 2014, 2015a):

- **Logistic Regression model (LR)**: it is a smooth regression model where the dependent variable is a categorical value. Usually, it is used for binary classification tasks, such as a Yes/No or a Dead/Alive discrimination. It was created by *Cox* (1958).
- **Naive Bayes model (NB)**: This model assumes that all inputs are conditionally independent, which is rarely true in some real-world applications. However, it tends to produce good results and works well with big data (*Zhang*, 2004).
- **Decision Trees**: Proposed by *Jr. and Johnson* (1959), decision trees are a non-parametric supervised learning method. The purpose is to create a model that predicts a target variable by learning mild decision rules deduced from the data features. These models are simple to understand and require little to no effort in preparing the data. However, the models often do not generalize well, thus creating over-complex trees and even biases if some classes dominate.

- **Random Forests model (RF)**: it is an ensemble learning method created by *Breiman* (2001). This model can be used for both classification and regression tasks. It builds many decision trees during their training, using a random selection of input features and training instances. The overall prediction is obtained by computing the majority class (classification) or average response (regression).

- **XGBoost model (XG)**: XG is short for Extreme Gradient Boosting. It denotes another ensemble learning method. Its purpose is to boost decision trees by maximizing their speed and performance. Tianqi Chen created it recently and it is based on the framework of *Friedman* (2001). This model offers three features:

  1. Gradient Boosting;
  2. Stochastic Gradient Boosting; and
  3. Regularized Gradient Boosting.

  The XG model can handle missing data values, which is an essential feature for real-world problems.

In recent years there has also been a surge of online learning methodologies using **ML** models (*Matos et al.*, 2018). Online learning is a relatively new area in the **ML** field and works in real-time, usually from data streams that contain data elements that are available over time (*Volkova*, 2012; *Bifet et al.*, 2018). As *Igelnik and Zurada* (2013) states, Online Learning is capable of outlasting its learning capabilities when compared with their offline counterparts since the offline algorithms tend to get outdated very quickly. The authors also suggest several Online Learning methods that can be used and how they should be used. In particular, there are currently some frameworks that deal with this type of learning by complementing some Offline Learning models with Online Learning capabilities such as:

- **Hoeffding Trees** – Which are anytime decision trees capable of learning from massive data streams. Being supported by the Hoeffding bound that is responsible for quantifying the number of observations making it able to estimate some statistics within a set of predictions (*Hulten et al.*, 2001; *Bifet et al.*, 2010, 2018).

- **OzaBoost** – It is known to be an Ensemble of models based on Bagging and Boosting. It has an effective generalization performance when compared to individual base models (*Oza*, 2005).

- **DecisionStump** – In essence, it consists of a Decision tree made of one level, and it includes an implementation in the MOA software (*Bifet et al.*, 2010, 2018).

2.9 Deep Learning

Deep Learning (DL) is a subfield of ML and Artificial Intelligence, and it consists of several Neural Networks (NN) architectures that contain a large number of hidden layers (*Bengio*, 2009). The hidden layers are responsible for most of the data processing effort and allow the model to solve complex tasks. Deep Learning has impacted the ML field, particularly after the 2010s, since several NN architectures, such as Convolutional Neural Networks (CNN), have won challenging competitions from the domains of Computer Vision and Natural Language Processing (*Vaswani et al.*, 2017).

According to *Hinton and Salakhutdinov* (2006), Deep Learning is often built in two phases. The first phase consists of initializing the model with unsupervised methods. This phase allows the model to learn the distribution of the data. The second phase consists in improving the model by employing a supervised learning algorithm (e.g., back-propagation) (*Jones*, 2017; *Goodfellow et al.*, 2016).

Several Deep Learning Structures are commonly used in Deep Learning and usually are used in Mobile Marketing, image recognition and voice recognition. Figure 4 presents a timeline of diverse Deep Learning architectures that have been proposed since the 1990s up to the present day. The NN types shown in the figure are: Recurrent Neural Network (RNN); Long short-term memory (LSTM); Convolutional Neural Networks (CNN); Deep Boltzmann Machine (DBM); Deep Stacking Network (DSN); Gated Recurrent Unit (GRU); and Deep FeedForward Neural network (DFNN).

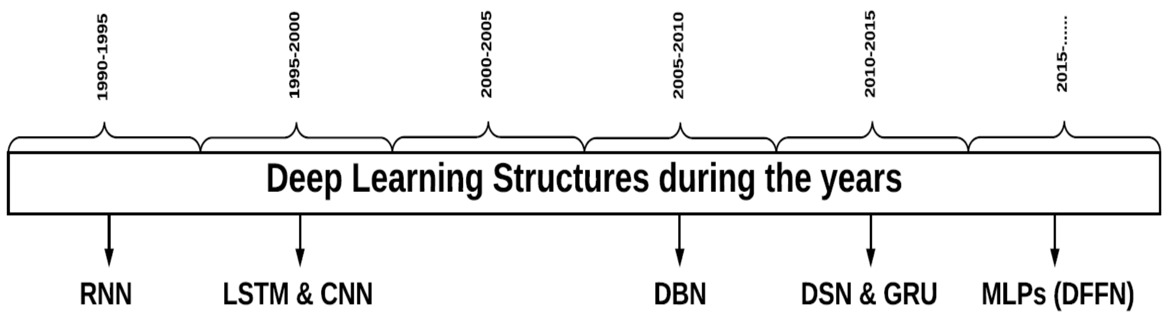

Figure 4: Structures of Deep Learning, adapted from *Jones* (2017) and *Goodfellow et al.* (2016)

2.9.1 *Deep Learning Structures*

This section discusses two Deep Learning models: **Deep Feedfoward Network (DFFN)** and **Convolutional Neural Networks (CNN)**.

Deep Feedforward Networks (DFFN)

DFFN, also known as **Multilayer Perceptrons (MLP)**, aims to optimize a function $f^{\star}$ by defining a mapping for $y = f(x;\theta)$ and learning the parameter values of θ, which results in a better approximation function. The **feedfoward** models allow the information to flow during the evaluation of x, regarding the computations used to define f obtaining the output of y. There are no feedback or recurrent connections, which means that the outputs do not return to themselves, nor are there cycles in the information flow. Networks that use recurrent connections are termed as **Recurrent Neural Network (RNN)**.

A DFFN is a modern version of the famous MLP used after the mid-1980s. The input layer contains the input values, and then several hidden layers process such values. The usage of chain layer structure is widely used in neural networks, making each "task" a layer. The chain's length determines the deepness of the model, which motivated the creation of the term of **Deep Learning**. The final layer is called the **output layer**, and it contains the model responses. Figure 5 represents an example of a DFFN with three inputs and N hidden layers, each with three processing nodes.

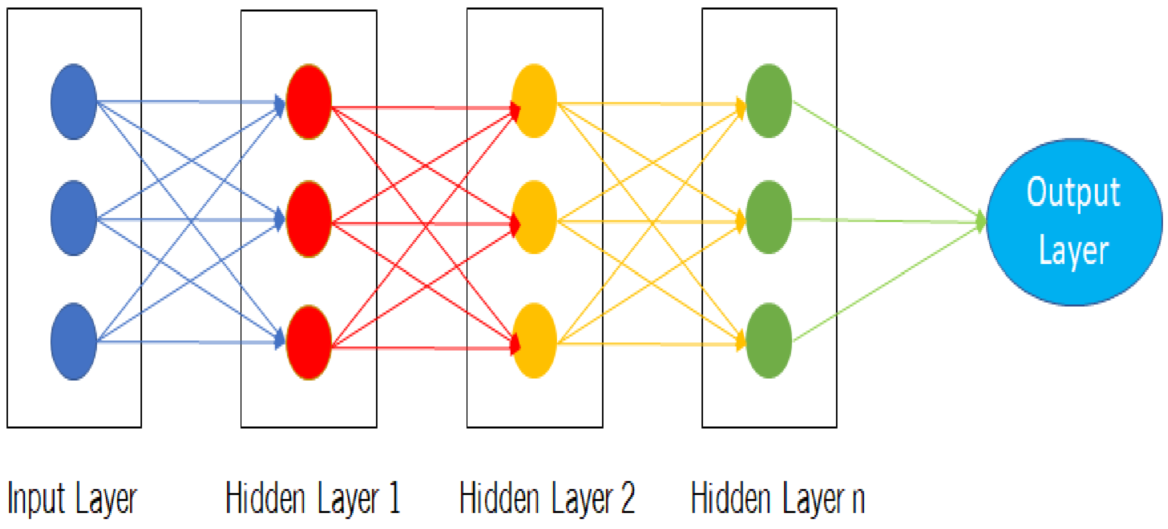

Figure 5: Example of a DFFN or MLP model

The DFFN is fundamental for the field of **ML** according to *Goodfellow et al.* (2016) since it forms the basis for several commercial applications. For example, the DFFN structure is part of the **CNN**, which is popular for object classification from images. An example applied to the Marketing domain is provided by *Zhang et al.* (2016), in which the authors proposed a Deep Learning model that included MLP layers to perform user response prediction.

Convolutional Networks (CNN)

The **CNN** were proposed by *LeCun et al.* (1989). These networks use a grid topology, which can be used to process time series and 2D images. The term "convolutional" is derived from the mathematical function that is used by these networks. The CNN is very effective in some applications, such as object detection in images (*Goodfellow et al.*, 2016).

Convolution is often associated with a pooling layer, a heuristic that reduces the dimensionality of the features obtained, typically by performing downsampling that keeps the most relevant information intact (e.g., max-pooling). After these steps, the network reuses the convolution function and the pooling heuristic to feed an MLP. The final output of the CNN model is an aggregation of the nodes that can identify the images' different characteristics. The CNN model is typically trained using a backpropagation method. Figure 6 presents an example of a CNN model.

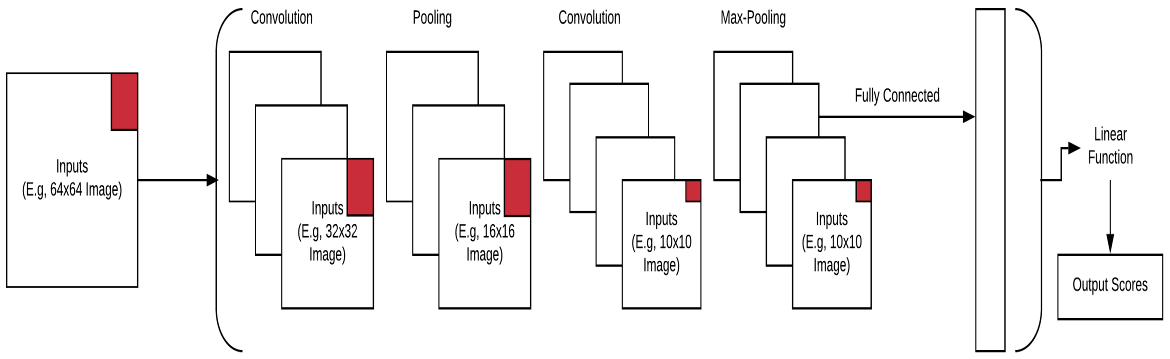
Convolution
Pooling
Convolution
Max-Pooling
Inputs (E.g, 64x64 Image)
Inputs (E.g, 32x32 Image)
Inputs (E.g, 16x16 Image)
Inputs (E.g, 10x10 Image)
Inputs (E.g, 10x10 Image)
Fully Connected
Linear Function
Output Scores

Figure 6: Example of a CNN model

2.9.2 *Deep Learning Regulators*

Several regulators can be used in Deep Learning networks. Their main objective is to improve the NN performance (e.g., accuracy, processing speed).

Goodfellow et al. (2016) describes some methods that can be used for regulating a Deep Learning network. The authors also address that the main issue, when working in the **Machine Learning (ML)** field, is **how** to make the algorithm capable of dealing with the new data and training them. The authors also define **regulators** as any modification made to the algorithm to make it more general and not reduce the error cost of the training. Therefore, several strategies can add more "constraints" to the model, set in terms of the memory allocated to the model's training or adding extra parameters to the objective function. The correct adjustment of these regulators and the parameters that define them can potentially increase the algorithm's performance when handling test data.

Goodfellow et al. (2016) presented several regulators that could be used with Deep Learning models. In this work, we highlight two regulators:

- **Early Stopping**: when training a neural network model using a high number of epochs, the training error might be largely reduced, but this could mean overfitting the data by memorizing it and losing generalization capability. One way to solve this issue is to split the training data into training and validation sets, using the former set to fit the model and the latter set to decide when to stop the training. This process is called **Early Stopping**, and it is widely used as a regulator in the Deep Learning field. One method to use Early Stopping is to act as a selector of hyperparameters, which came to be very efficient since we can consider the number of training steps (**epochs**) a hyperparameter. Often, most hyperparameters tend to have a U curvature in terms of performance, meaning that if the validation error rises, then the training algorithm should be stopped.

Early Stopping is considered a non-obstructive form of Deep Learning regularization since it does not require significant changes in its architecture, training, and objective function. This means that Early Stopping is simple to implement and does not damage the model itself. Early Stopping can also be combined with other forms of regularization, such as Dropout.

- **Dropout**: Proposed by *Srivastava et al.* (2014), Dropout is largely adopted in Deep Learning. The method consists of randomly dropping out neural units and connections. The user defines a percentage of Dropout (e.g., 30%) that is applied to the whole neural network or just to a particular layer.

2.9.3 *Activation Functions*

When building a **Deep Learning (DL)** model, it is necessary to focus on the layers that constitute it. Depending on the problem, certain activation functions can be used to improve the performance and convergence of the model. These same functions define how the weighted sum of the input is transformed into an output from node to node between the several layers.

A **Deep Learning (DL)** network consists of three types of layers: Input Layers, which are layers that receive raw input; Hidden Layers that receive input from the other layers and send as output to other layers and, finally, the output layers that execute and launch the model forecasts.

Typically, all hidden layers use the same activation function, and the output layer uses a different activation function that depends on the forecasting goal.

Several activation functions are used in **Deep Learning (DL)** networks. In this work, we focus on some of them (*Li et al.*, 2021):

- **Sigmoid** – The sigmoid activation function, also known as the logistic function, is widely used for the output layer in binary classification purposes. This function takes any real value as input and outputs values within the range $[0; 1]$. The higher the input, the more positive will the output be (closer to 1) and vice versa. The sigmoid activation function is calculated as follows:

$$Sigmoid = \frac{1}{(1 + e^{-x})} \tag{1}$$

- **Softmax** – The Softmax activation function is a generalization of the sigmoid function used for multiple dimensions. Similarly to the sigmoid function, it is widely used in the output layer when the results should correspond to a probability distribution (e.g., Multi-classification tasks). The softmax function is calculated as follows:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \qquad (2)$$

- **ReLU** – The ReLU activation function is perhaps the most common function used for hidden layers. It is less sensitive to vanishing gradients that prevent Deep Learning models from being trained, although it can experience other problems named "dead" units. These "dead" units represent some ReLU Neurons that essentially die for all inputs and remain inactive regardless of the supplied input. In larger **Deep Learning (DL)** the ReLU function performs faster and avoid backpropagation errors when compared with other activation functions (e.g., Sigmoid). The ReLU function is calculated as follows:

  $$ReLU = max(0, x) \qquad (3)$$

  Therefore, if the input value (x) is negative, then a value of 0 is returned. Otherwise, the value x is returned.

2.9.4 *Optimization Functions*

Deep Learning (DL) models use optimization functions to train and improve the model, depending on the objective and loss function chosen. These functions are responsible to change the attributes of the neural network such as weights and learning rate to reduce some specific loss function.

There are several optimization functions with specific purposes depending on the task that is being modulated. Some known optimization functions are (*Chollet*, 2017):

- **Stochastic Gradient Descent (SGD)** – It is a variant of Gradient Descent. The SGD updates the model's parameters more frequently. Thus, the model parameters are altered after each computation of the loss function for each training example. Some advantages are that frequent updates make the model converge in less time and require less memory since it does not need to store the loss function values. However, there is a high variance in the model parameters, disrupting the loss even after achieving the global minima, and the learning rate needs to be reduced at a plodding pace (*Sutskever et al.*, 2013).

- **Adam** – Adam is an efficient optimization function for stochastic optimization. This algorithm only requires the first-order gradients while using little memory requirements. Adam also computes the individual adaptive learning rates for different parameters deriving from estimates of first and second moments of the gradients. As such it combines the advantages of the RMSProp and Adagrad algoritms (*Kingma and Ba*, 2015).

- **AdaDelta** – Adadelta is a more robust extension of Adagrad. Adadelta is a stochastic gradient descent method based on adaptive learning rates and addresses two drawbacks: the continual decay of learning rates during the training phase and the need to have a selected global learning rate (*Zeiler*, 2012).

2.9.5 *Loss Functions*

When training a Deep Learning model, several sets of loss functions can be used to maximize the model quality. The choice of a loss function depends on the reality in which the model is inserted (e.g., Classification, Regression, AutoEncoding), and its result depends on the chosen optimization algorithm (e.g., Adam, SGD).

In this work, we are interested mostly in binary classification. As such, we have selected the following loss functions (*Chollet*, 2017):

- **Binary Crossentropy** - Binary Crossentropy is a probablistic loss function that computes the cross-entropy loss between true labels and predicted labels. It is widely used when handling binary classification tasks.

- **Categorical Crossentropy** – Categorical Crossentropy is also a probablistic loss function similar to the Binary Crossentropy, however, this loss requires two or more label classes represented in a One Hot representation.

2.10 Preprocessing Methods

There are several data preprocessing techniques that transform the raw data into a format that is easier to be understood by the data analyst or to be learned by the ML algorithm. In this section, we mainly discuss categorical transformation methods. Categorical attributes are the most common ones in the Mobile Marketing Performance domain. Moreover, several ML algorithms (e.g., Logistic Regression) can only process numerical inputs. Thus there is a need to apply a categorical to numeric transform.

The most popular categorical transform is termed **One-Hot-Encoding** and it encodes categorical values into a binary vector with L levels, where L is the attribute's cardinality. For instance, the attribute color with $L = 3$ levels {"blue","red","yellow"} would be transformed into: "blue" → (1,0,0); "red" → (0,1,0); and "yellow" → (0,0,1). Since this transform can be applied to virtually any domain, it is the default encoding assumed by most Data Mining tools. It is also widely adopted, including in the Mobile Marketing domain, such as used by *Du et al.* (2016). However, this transform presents a drawback, since it heavily increases the number of binary input features and computational effort (in terms of memory and processing power) when applied to attributes with high cardinality.

Another categorical to numeric transform method is the **Inverse Document Frequency (IDF)**, which was proposed by *Campos et al.* (2016). **IDF** is a data transformation process where a categorical value is encoded according to the following equation:

$$IDF(t) = ln(\frac{N}{f_t}) \tag{4}$$

where N denotes the total number of instances (values), and f_t is the number of occurrences of level t in the training data. When using this transform, a closer numeric value to 0 means that the level is persistent in the data. The higher the value, the less frequent the level is, with the less frequent levels being grouped closer.

2.11 Balancing Methods

In classification tasks, the target class is often unbalanced, with one class outnumbering the others. For instance, the CVR predictive task of the Mobile Performance Marketing domain is heavily unbalanced since only a tiny fraction (e.g., around 1%) of the users' clicks result in a conversion.

When using such unbalanced data, the ML algorithm will quickly learn the majority class but often will produce a poor predictive performance on infrequent classes. This issue can be solved by changing the ML learning function or the training data distribution. The former method is often dependent on the ML algorithm. Also, it may be difficult to set the appropriate cost function correctly. In this work, we focus on training sampling balancing methods, which can be applied to any ML classifier. These methods can be classified into two main types (*Chawla et al.*, 2002; *Batista et al.*, 2004): undersampling and oversampling.

Undersampling involves reducing the majority class examples by using a sampling method, typically using a uniform distribution. The training set size will be reduced since a large portion of the majority class samples are discarded, such that the data becomes balanced (*Batista et al.*, 2004).

Oversampling performs the augmentation of the training dataset. The standard approach involves a random sampling with replacement of the minority classes, such that the data becomes balanced. A more sophisticated approach is provided by the SMOTE technique (*Chawla et al.*, 2002), which creates new synthetic data by randomly defining data points close to the actual minority samples.

2.12 Programming Languages for Data Science and Machine Learning

With the advent of information systems and computing technology, several programming languages were proposed to solve and create systems to be applied in several fields. Some programming languages are more suited for scalability of the system (e.g., Java, C, C++, Python), others are more suited for web applications (e.g., Java and Javascript), some are used mostly for analyzing data and apply Machine Learning models (e.g., R and Python) and others are used for system deployment (e.g., Python). In this PhD work, we assumed will delve more into the *Python* and *R* programming languages since they are widely used for Data Science and Machine Learning purposes. These tools were used to develop a prototype of the artifact requested by the Mobile Marketing company.

2.12.1 *R*

R (*R Core Team*, 2016) is both a programming language and a statistical and computing environment. The R tool was created by the Bell Laboratories and includes a large variety of statistical methods. The tool can also be expanded through the installation of packages. this PhD work, we used several packages, namely rminer, DMwR, ROSE and doParallel.

The *rminer* package, proposed by *Cortez* (2016), offers a coherent set of functions to perform classification and regression tasks and evaluate the respective predictive performances. Some of the implemented classification methods based on this package.

The *DMwR* (*Torgo*, 2010) and *ROSE* (*Lunardon et al.*, 2014) packages offer several forms to apply oversampling and undersampling of the data. These packages are valuable for the PhD project since the number of positive cases (the Conversions) is very small.

The R language also offers a form of parallelization of the system by by using the *doParallel* package. Proposed by *Corporation and Weston* (2020), this package provides functions for parallel execution of R code on machines with multiple cores or processors or multiple computers. This was used by the data collection system (Fig. 9) and facilitated the concurrent collection of redirect and sales records, their treatment and storage.

2.12.2 *Python*

Python is a high-level, general purposed and interpreted programming language. This programming tool offers both automatic memory and disk management. It supports multiple programming paradigms and it has a large comprehensive standard Library (*Rossum*, 1995).

Python is built as an open source environment and it also offers a community based development model. Nowadays, it is considered one of the top programming languages in the field of science. In this work, we focus on some of the more relevant libraries, namely: Keras, Rpy2, SciPy and Scikit-learn.

Keras is a high level Deep Learning tool and is made in Python. It is based on the platforms *Theano*, *CNTK* and *Tensorflow*. Keras allows developers to quickly implement a prototype. It supports both conventional and recurrent networks, as well as a mixing of both, that runs homogeneously in both CPU and GPU processors (*Chollet*, 2017).

Rpy2 is a Python interface for running R code. This means that the R code will be represented as classes inside the Python system, allowing for the migration of R based code to Python and vice-versa (*Gautier*, 2016). This library is particularly valuable for this project, since the IDSS is implemented using both R code and Python code.

SciPy is an open-source library for Python. It is used for scientific computing and technical computing. It includes several other models that provide support for scientific tasks (e.g., optimization, linear algebra, integration, interpolation, special functions, signal and image processing) and it was made by *Jones et al.* (2001). This library contains two main modules: **NumPy** and **Sparse**. Although NumPy is another library, SciPy extends the Numpy capabilities by using the *pandas* and *Matplotlib* modules. The NumPy library offers support for large, multi-dimensional arrays and matrices, and it provides a large collection of high-level mathematical functions that work upon these arrays (*Oliphant*, 2006). Regarding the Sparse module, it provides methods that compress data into sparse matrices. This allows to save memory when performing Deep Learning tasks with Big Data characteristics (*Jones et al.*, 2001).

Scikit-learn is a library in Python that implements several **ML** techniques (e.g., Supervised Learning, Unsupervised Learning, Classification and Regression) (*Pedregosa et al.*, 2011). In this PhD project, we use scikit-learn to compare the Deep Learning models with a base **LR** model, in order to verify if Deep Learning is competitive when applied to the Mobile Marketing field.

2.13 Machine Learning Evaluation Metrics

When building **Machine Learning (ML)** models, it is necessary to assess and evaluate their performance. Depending on the value obtained by a specific metric (or metrics), it can be an indicator that the model responds/predicts/distinguishes well or poorly to certain realities (e.g., Classification , Time-Series Prediction, Regression). With the metrics employed, there is always the possibility of choosing specific models over others.

In this work, the following classification metrics were considered to evaluate the Machine Learning models predictive performances:

- **Receiver Operating Characteristic (ROC)** – ROC is a popular analysis used to evaluate classifier output quality in a binary classification scenario. ROC curves typically represent the true positive rate on the Y-axis and a false positive rate on the X-axis. The model should aim to the top left corner of the plot, which is the ideal point (*Fawcett*, 2006).

- **Area Under Curve (AUC)** – AUC is obtained by measuring the two-dimensional area underneath the entire Receiver Operating Characteristic (ROC) curve and it is situated between 0 to 1. Often, the quality of the AUC values are interpreted as: 0.5 – performance of a random classifier; 0.60 - reasonable; 0.70 – good; 0.80 – very good; 0.90 – excellent; and 1 – perfect.

- **Confusion Matrix** – The Confusion Matrix, also known as an error matrix, is used to evaluate the accuracy of classification by outputting a specific table layout that assesses the performance of an algorithm. Each row of the matrix describes an actual class, while each column depicts the predicted labels, or vice versa, depending on the programming language and configurations.

- **Accuracy** – Accuracy is the proportion of true results among the total of cases that were examined. It is usually used in classification problems, however it should be noted that the datasets should be well balanced. This metric ranges between 0% to 100% and it is computed by:

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)} \tag{5}$$

  where TP, TN, FP and FN denote the number of True Positives, True Negatives, False Positives and False Negatives.

- **Precision** – Precision is a metric that calculates the ratio of correctly predicted positive observations to the total predicted positive observations ranging from 0% to 100% and it is computed by:

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

- **Recall** – Recall is a sensitivity metric based on the ratio of correctly predicted positive observations to the all observations in actual class. This metric ranges between 0% to 100% and it is computed by:

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

- **F1 Score** – F1 score, also known as balanced F-score or F-measure, can be interpreted as a weighted average of precision and recall, where an F1 score reaches its best value at one and the worst score at 0. The contribution of either precision and recall for the F1 score is equal. The formula for the F1 score is:

$$F1 = 2 \times \frac{(precision \times recall)}{(precision + recall)} \tag{8}$$

- **LIFT** – In machine learning, the LIFT curve measures the performance of a targeting model at predicting/classifying cases as having an enhanced response (concerning the population as a whole), measured against a random choice targeting model. If the targeting model performs well, then the target's response is much better than the average. LIFT is simply the ratio of these values: target response divided by the average response, computed as follows:

$$LIFT = \frac{P(A \cap B)}{P(A) \times P(B)} \tag{9}$$

2.14 Mobile Performance Marketing

The Internet worldwide expansion has created huge marketing opportunities, such as improving marketing campaigns for e-commerce applications or obtaining consumer profiles from Facebook (*Gubela et al.*, 2019; *Chen et al.*, 2019). In particular, the massive adoption of 4G or Wi-Fi connected smartphones and tablets has increased the value of mobile performance marketing.

Performance Marketing returns a monetary compensation only when an ad performs well. In the Mobile domain, this is achieved by using a **DSP**, which is used by publishers and advertisers. Publishers are the website owners and they are responsible for generating content that is capable of attracting a vast audience. To finance their content (e.g., online games, news) publishers use ads with dynamic links in their web pages. Once a user clicks on an ad, the DSP selects an advertisement campaign, redirecting her/him to an advertiser site. When analyzing a user, the DSP can consider several aspects like:

- Is this a premium user?
- What particular aspects do I know about the user?
- What advertiser is best suited to appear?

If the website is not committed to any advertiser in particular, the publisher can submit this space to an open market, by connecting into **Supply-Side Platforms (SSP)**, which are used for monitoring the publisher ad inventory. The **DSP**, similar to stock brokers, can buy the space and then offer conditions based on the ad preferences (e.g., ads specific to males withing a range of age) usually named a pre-cached bid that is known to be the fastest way of filling an empty ad space by a **DSP**. If there is no pre-cached bid system associated with the ad-platform, the publisher can open an auction to the available **DSP** and this is done in less than 10 milliseconds. After this auction is complete, either by normal bidding or by pre-cached bids, the **DSP** sends instructions to the ad-exchange platform, that subsequently finds its way back to the publisher. This process is also known as **RTB**, and usually happens in less than a second (*Barros*, 2017; *Bureau*, 2014; *Du et al.*, 2016).

In this work, we had access to a world global **DSP**, that is supported by a private company called *OLAmobile*. The DSP is responsible for the RTB process, making the connection between publishers, advertisers and users, as shown in Figure 7. The *OLAmobile* company usually operates with more than one hundred thousand publishers and more than two hundred markets, with more than two thousand campaigns occurring at the same time (Figure 8). The DSP analyzed in this work generates millions of user clicks per hour, thus any analytics need to be processed using Big Data storage technology.

The analysed DSP acts as a broker, matching user profiles to ads, thus linking user traffic, coming from publishers (e.g., news site, game app) to advertisers. And, if there is a conversion (e.g., product sale), the DSP facilitates the cash flows, returning a portion of the advertiser's revenue to the publishers. Thus, publishers can fund their services by adhering to a DSP. Publishers typically attract a vast audience of users, which need to click a dynamic link prior to access the publisher content. This dynamic link is managed automatically by the DSP. Once it is clicked, the DSP selects a marketing campaign, redirecting the user to a specific ad and advertiser. All ad clicks and sales generate data events: redirects, when users click the dynamic link; and sales, when there is a conversion. A critical aspect of the DSP big data system is the prediction of the user Conversion Rate (CVR), which involves estimating if there will be a sale when a user clicks and views an advertisement. In this PhD, we use data-driven ML models, trained with redirect and sales events, to perform mobile user CVR prediction.

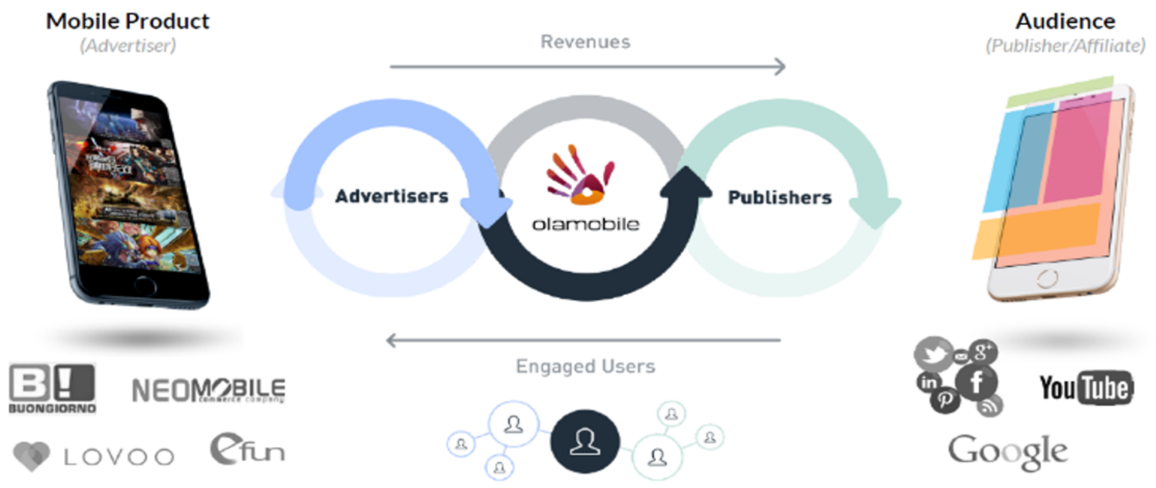

Figure 7: Description of process of RTB retrieved from *OLAmobile* (2011)

2.14.1 *User Click-Through Rate Prediction*

Regarding **Click Through Rate (CTR)** prediction, the work of *Zhang et al.* (2016) is focused on Deep Learning techniques with multi-categorical data. The authors introduced two models for Deep Learning for **CTR**: the Factorization Machine supported Neural Network (FNN) and Sampling-based Neural Network (SNN). The first is responsible for efficiently reducing sparse features into more continuous ones and the SNN model is a deep neural network powered by a sampling-based restriction (Boltzmann) with the proposed negative sampling method. The authors used *One-Hot-Encoding* preprocessing method to handle categorical input features.

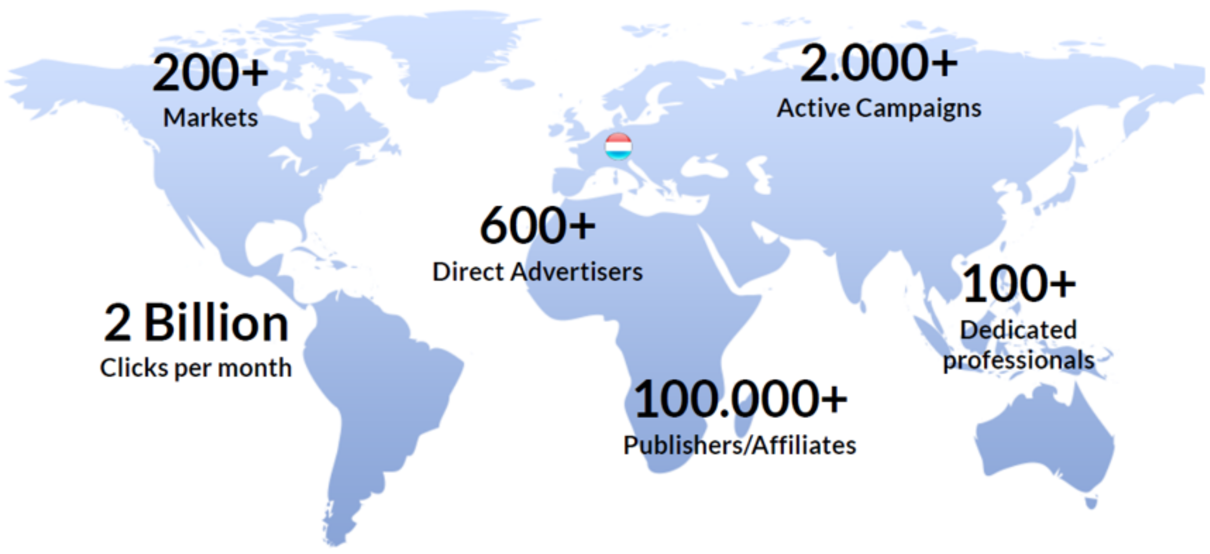

Figure 8: Characteristics of the mobile market from *OLAmobile* (2011)

Another relevant work in the field of **CTR** prediction is the work of *Chen et al.* (2009), aiming at the creation of a solution based on behavioral targeting. The purpose was to predict the **CTR** using a Hadoop MapReduce framework and the linear Poisson regression models. The authors started with a month of raw data, with the size of 30 TB, reducing it to 3 TB of data to conduct the experiments, that were later aggregated by using user cookie information. *Chen et al.* (2009) also conducted experiments with vector feature selection, in order to test the scalability of their model with different sliding window sizes and they were able to evaluate the models with **Receiver Operating Characteristic (ROC)**.

Wu et al. (2012) created an artifact for the competition in the National Taiwan University KDD cup and their main goal was to predict the click through rate of ads on the platform called *Tencent*, which is a database engine widely used in Asia. The authors also exploited several models and approaches, from regression to ranking and tested these approaches on two stages, one for private testing and the other for public testing. *Wu et al.* (2012) also blended several techniques to factorize, rank and apply the regression into their models.

Another study in this field is the work of *Guo et al.* (2018), which proposed the usage of an end-to-end wide Deep Learning framework for CTR prediction. For this, the authors developed a topology that used Factorization Machines for their CNN networks. Their approach was based on the use of embedding layers, which are capable of finding relationships between variables and their dimensional proportions (e.g., One-Hot encoding). After processing the data in these layers, the authors decided to use a Factorization Machine (FM) layer, which is similar to the work of *Zhang et al.* (2016). For these experiments, *Guo et al.* (2018) used offline models that were applied to an online environment, which was the *Huawei App Market*, obtaining a 10% improvement over the Logistic Regression model already present in that platform.

Guo et al. (2021) state that most deep CTR models follow an Embedding & Feature Interaction paradigm, and the majority of methods focuses on designing network architectures to capture feature interactions better. In contrast, feature embedding, especially for numerical features, has been overlooked. Therefore, the authors propose a novel embedding learning framework for numerical features in CTR prediction (AutoDis) with high model capacity, end-to-end training and unique representation properties preserved in order to learn global knowledge from the perspective of the field with a manageable number of parameters. This framework achieved interesting results in CTR when compared with other Deep Learning models.

2.14.2 *User Conversion Rate Prediction*

Regarding **CVR** prediction, one relevant work was performed by *Du et al.* (2016), which focuses in learning and/or predicting user behavior while using static features, such as location, software and hour when the purchase was made. The authors worked with data from the same DSP that is analyzed in this work, although using different types of features. As for the predictive analytics, they explored Naive Bayes, Logistic Regression and Random Forests.

Yang and Lu (2016) also studied **CVR** prediction. The authors applied an extension of matrix factorization called Bayesian Heteroscedastic Matrix Factorization (BHMF) to the **DSP** area. The authors also had to deal with unbalanced data (below 1%). The One-Hot-Encoding transform was adopted. The study included an extension of *Markov Chain Monte Carlo* method that used importance weights and random variables.

The work of *Yoshikawa and Imai* (2018) considered the time delay between the ad click and the actual conversion as an important variable in the non-parametric delayed feedback model (NoDeF). The proposed method works as an ensemble of two models, in which the first is responsible for representing the actual time delay between the conversion and ad click, and the second one is the predictive conversion model.

Wen et al. (2020) observed that users always take several purchase-related actions after clicking. As such, they proposed a novel idea of post-click behavior decomposition. Specifically, disjoint purchase-related Deterministic Action (DAction) and Other Action (OAction) that were inserted between clicks and purchases in parallel, forming a novel user sequential behavior graph “impression - click - D(O)Action - purchase”. The graph model enabled the authors to leverage all the impression samples over the entire space and also devised a novel deep recommendation model named “Elaborated Entire Space Supervised Multi-task Model” (*ESM2*).

2.15 Summary

This chapter presents the background necessary for this work, including research topics, concepts and state of the art works applied to the mobile marketing domain.

During this work we have come across with several articles regarding **ML**, **CTR** and **CVR** in the field of Mobile Marketing platforms. Table 1 presents the articles that are more aligned with the focus of this PhD research project alongside the goals, datasets, models, methods and metrics used.

Given the interest in online advertising, several works have addressed the prediction of user **CTR** or **CVR**, mostly using offline (batch learning) linear models, such as **LR** as seen in the work of *Du et al.* (2016). Recently there has been more flexible offline **ML** models, such the **RF**, that were using in the work of *Du et al.* (2016), Gradient Boosting Decision Trees, in *Zhang et al.* (2014) and in *Lu et al.* (2017) and also works using techniques from the Deep Learning (DL) field (*Zhang et al.*, 2016; *Guo et al.*, 2018; *Wen et al.*, 2020; *Guo et al.*, 2021).

However, these **ML** models tend to consider only the prediction performance and give almost no attention to the computational effort that is used to generate the learning methods, nor does consider the concept drift that occurs when dealing with data streams, as referred in Section 2.7. For instance, in terms of complexity, Deep Learning is much more complex than the LR method and it only improves **AUC** metric by 0.1% (*Zhang et al.*, 2016; *Wen et al.*, 2020; *Guo et al.*, 2021). Since **DSP** generates big data, we assume here that the computational effort is a relevant aspect when comparing the **ML** models and recently some studies have started to consider the inference time in an online learning environment as an important aspect to consider when deploying and testing these models (*Wen et al.*, 2020; *Guo et al.*, 2021).

Regarding selection of the models, including preprocessing methods, transformations and feature selection, training algorithms to be used, to be a relevant in the domain of **CTR** and **CTR**, we have also to consider that these domains are highly imbalanced (e.g., only 1% of the generated events turn into a sale). Thus we believe that balancing the training data (e.g., undersampling as seen in *Lunardon et al.* 2014 or SMOTE *Chawla et al.* 2002) could improve the results of the model, although this aspect has been scarcely studied in this domain.

Another issue is that most data concerning **CTR** and **CTR** are categorical and several of these attributes can contain large cardinalities, with the possibility of reaching thousands of values. The state of the art works adopted the raw encoding (*Zhang et al.*, 2014) or the one-hot-encoding (*Du et al.*, 2016; *Zhang et al.*, 2016; *Guo et al.*, 2018) and thus distinct categorical values are scarcely compared. For example, the one-hot-encoding is a popular Data Mining transformation but it has severe drawbacks, since it increases the computational memory effort and leads to an unfeasible use of use of some **ML** models.

We also share the opinion of the *Amado et al.* (2018) that states that big data is not yet aligned with cutting edge technologies that could aid marketing fields. This lack of alignment occurs because technologies that try to connect big data to marketing are still in their embryonic stage (e.g., lack of computational efficiency, speed of storage, hardware). This can lead to new possibilities when dealing with big data environments and using Data Mining approaches, creating many new opportunities for future research in this area.

Table 1: Summary of the state-of-the-art works

Study	**Goal**	**Model/s**	**Metrics**	**Source**	**LT**[l]	**PP**[p]
Chen et al. (2009)	Use of behavioral data to predict CTR	Linear Algorithm	LIFT[h]; ROC	Yahoo database	Offline	n.d.[z]
Wu et al. (2012)	Creation of an ensemble model used to rank advertisements	Ensembles; NB[f]; LR[a]; RR; SVR; RLR; RN; CRR; RMF; RankMF	AUC[g]	Tesent database	Offline	One-Hot; IDF; Tf-IDF
Power (2014)	Insight for data analysis when dealing with big data	––	––	––	––	n.d.[z]
Moro et al. (2014)	Improve the rate of success of a deposit on a Bank	LR[a]; DT[b]; NN[d]; SVM[e]	AUC[g]	Phone Records of a Bank	Offline	Label Encoding
Arnott and Pervan (2014)	Investigate the changes in the DSS field	––	––	Literature Review	––	n.d.[z]
Moro et al. (2015a)	Choose the most profitable campaigns to the right consumers by predicting their behavior	LR[a]; DT[b]; NN[d]; SVM[e]	AUC[g] ALIFT[i]	Telemarketing bank campaigns	Offline	n.d.[z]
David et al. (2015)	Techniques and strategies for marketing and big data analytics	––	––	Literature Review and Professional Opinions	––	n.d.[z]

Study	Goal	Model/s	Metrics	Source	LT[l]	PP[p]
Deng et al. (2015)	Developing Big Data Mobile Marketing analytics and advertising recommendation framework	ML techniques Clustering	Euclidean Distance	User information and interests Products from local Businesses	Offline	n.d.[z]
Du et al. (2016)	Insight on several approaches and their comparison either for learning and/or predicting user behavior while using static features	LR[a]; NB[f]; RF[c]	AUC[g]	Ads from mobile applications	Offline	One-Hot
Zhang et al. (2016)	Training deep neural networks (DNNs) to predict users' ad click response based on multi-field categorical features.	LR[a]; FM; FNN; SNN-DAE; SNN-RBM	AUC[g]	iPinYou dataset	Offline	One-Hot
Yang and Lu (2016)	An extension of a matrix factorization (BHMF)	*Trans-RWM*	AUC[g]	Brightroll	Offline	Word to Vector
Yoshikawa and Imai (2018)	A creation of an ensemble (NoDeF)	Ensemble Techniques	Log loss; ACC; AUC[g];	Synthetic Dataset	Offline	One-Hot
Wen et al. (2020)	A Multi-Task Modeling Post-Click Behavior Decomposition for CVR Prediction	Deep Ensemble Embedding Models	AUC	Real-world e-commerce dataset	Offline and Online	embedding techniques

Study	Goal	Model/s	Metrics	Source	LT[l]	PP[p]
Guo et al. (2021)	The creation of an embedding learning framework for numerical features in CTR Prediction	Automatic Platform (Autodis)	AUC	Two public and one industrial datasets	Offline and Online	embedding techniques

a – LR: Logistic Regression; b – DT: Decision Tree; c – RF: Random Forest; d – NN: Neural Network; e – SVM: Support Vector Machines; f – NB: Naive Bayes; g – AUC: Area Under Curve; h – LIFT: Lift Curve; i – ALIFT: Cumulative Lift Curve; j – MAPE: Mean Absolute Percentage Error; k – ACC: Accuracy; l – Learning Type; p – PreProcessing; z – Not disclosed

3

METHODS, EXPERIMENTS AND RESULTS

This chapter presents the main methods, experiments and results obtained throughout this PhD work (following a chronological order):

- The first set of experiments are related with an initial comparison study of Machine Learning (ML) approaches for binary user Conversion Rate (CVR) prediction. The associated work was published in the International Conference on Intelligent Systems (IntS) (*Matos et al.*, 2018) (Section 3.1).
- The second set of experiments focused on the usage of Deep Learning models for CVR prediction and it was accepted in a **CORE Rank A**[1] conference, International Joint Conference on Neural Networks (IJCNN), (*Matos et al.*, 2019a) (Section 3.2).
- The third set of experiments target the application of Deep Learning models for ordinal classification and it was accepted in International Conference on Intelligent Data Engineering and Automated Learning (IDEAL) (*Matos et al.*, 2019b) (Section 3.3).
- The last set of experiments are described in an article that was submitted to an International Journal (*Matos et al.*) (Section 3.4). The article addresses an Intelligent Decision Support System (IDSS) to automatically select mobile marketing campaigns for users based on a binary deep Multilayer Perceptrons (MLP) model.

Whenever possible, in this chapter we maintained the original text of the published papers. It should be noted that since the papers were published in different scientific venues, at different publication years, there might be slight changes in the notation or terms used.

3.1 A Comparison of Data-Driven Approaches for Mobile Marketing User Conversion Prediction

Mobile performance marketing is growing due to the widespread usage of mobile devices. Several mobile advertising commercial platforms have been created, in what is known as Demand-Side Platform

1 See: `http://portal.core.edu.au/conf-ranks/685/`.

(DSP). The DSP acts as a broker, matching user profiles to ads, thus linking user traffic, coming from publishers (e.g., news site, game app) to advertisers. And, if there is a conversion (e.g., product sale), the DSP facilitates the cash flows, returning a portion of the advertiser's revenue to the publishers.

In this set of experiments, we perform an exploratory study of user mobile Conversion Rate (CVR) prediction using recent big data from a global mobile marketing company. Firstly, we designed a stream processing engine to collect sampled mobile marketing data. Then, we executed a large set of CVR tests, under a two-stage experimental procedure that considered a rolling window evaluation. Several preprocessing and Machine Learning combinations were analyzed using a preliminary set of data. Next, the selected combinations were tested on a larger set of unseen datasets. Interesting classification performances were achieved, with some learning models (e.g., XGboost) requiring a reduced computational effort.

3.1.1 *Stream Engine and Collected Data*

As previously described, publishers can fund their activities by adhering to a DSP and putting a dynamic link in their web pages or apps. Once it is clicked, the DSP selects a marketing campaign, redirecting the user to a specific ad and advertiser. Two main DSP data events are generated: redirects, when users click on the dynamic link; and sales, when there is a conversion. All events are stored at the DSP data center, which receives millions of redirects and thousands of sales per hour. The DSP provided us with a secure web service (https) where it was possible to request NR redirects or NS sales from the data center. In this work, we had access to an Intel Xeon 1.70GHz server with 56 cores and 2TB of disk. This server has limited capabilities when compared with the data center and thus we worked with sampled data. We designed a stream processing engine (Figure 9) using the R tool (*R Core Team*, 2016). The engine sets K cores for requesting redirects and sales. After receiving the stream (in JSON format), each core sleeps for SR or SS seconds and then repeats the request (asking for more data). The received streams are sent to another layer of cores, that filter the data according to some of its attribute values. The filtered streams are then stored (first in, first out - FIFO order) in three files: redirects; sales; and an event log, used for monitoring the data collection. These files were stored using MongoDB, a fast NoSQL JSON database system (*Banker*, 2011).

Table 2 describes the stream collection parameters and resulting datasets. The **Traffic** column distinguishes two main event types: TEST – initial DSP testing mode, used to measure campaign performance; and BEST – with best product campaigns that have obtained a minimal TEST performance and that corresponds to most traffic. The last two columns denote the number of redirects that produced sale (Y_{yes}) and no sale events (Y_{no}).

Due to web service limitations it was not possible to retrieve all redirects for the shorter duration datasets. Thus, our ratio of collected sales $Y_{\text{yes}}/(Y_{\text{no}} + Y_{\text{yes}})$ is often higher than the real DSP ratio, ranging from 2.1% to 34.4% (BEST) and 0.2% to 20.4% (TEST). This issue is handled by setting two data variants: **collected**, with all stored events, and **realistic**, with a sample of the collected data

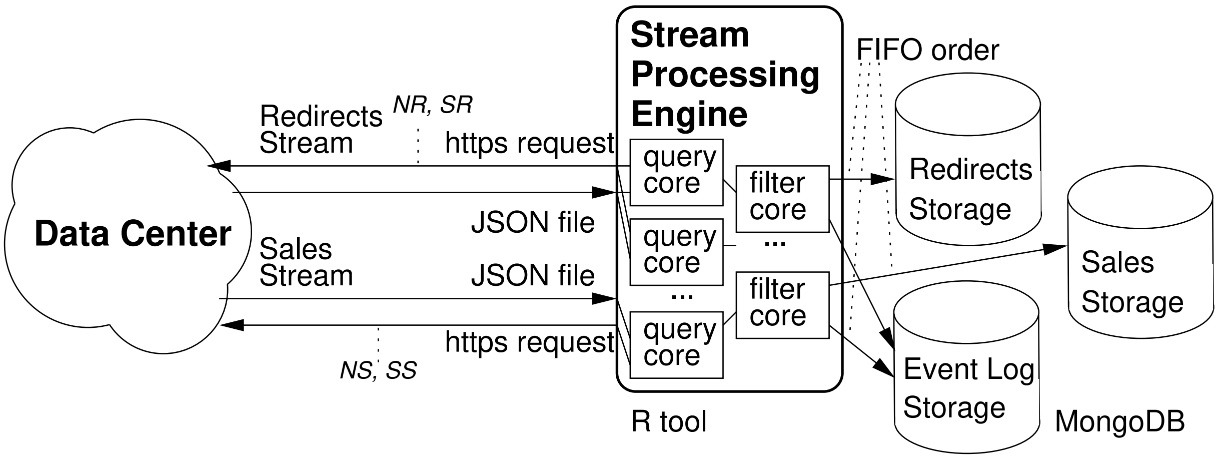

Figure 9: Scheme of the developed stream processing engine

Table 2: Summary of the stream engine setup and collected datasets

Traffic	Duration	Start	NR	NS	SR	SS	K	Y_{no}	Y_{yes}
BEST	30 min.	14/09/2017 16:12	7,500	2,000	0	0	5	235,069	9,401
	1 hour	28/11/2017 19:29	5,000	1,000	0	0	5	229.707	4.847
	1 day	16/11/2017 12:55	500	500	60	60	2	227.687	119,388
	1 week	07/12/2017 15:23	70	50	150	150	2	180,629	80,775
TEST	30 min.	28/09/2017 09:29	10,000	2.000	0	0	5	85,312	167
	1 hour	22/01/2018 18:10	7,500	1.000	0	0	5	217.102	765
	1 day	22/01/2018 18:10	5,000	500	60	60	5	216,369	1,242
	1 week	17/12/2017 22:25	200	50	150	150	2	98,172	25,131

such that the overall sales ratio is 1% for BEST and 0.5% for TEST. Table 3 lists the input categorical attributes and output target (last row) provided by the DSP. The attributes are defined in terms of their context (related with whom) and description (including the number of levels and example values).

Table 3: Description of the data attributes

Context	Attribute	Description
user	country	user country: 192 to 216 levels (e.g., Russia, Spain, Brazil)
	region	region of the country: 22 to 24 levels (e.g., Asia, Europe)
	browser	browser name: 13 to 14 levels (e.g., Chrome, Safari)
	operator	mobile carrier or WiFi: 377 to 437 levels (e.g., Vodafone)
advertiser	vertical	ad type: 3 to 5 levels (e.g., video, mainstream, dating)
	campaign	ad product identification: 797 to 1564 numeric levels
	special	smart link or special offer: 451 to 1271 levels
publisher	account	publisher type: 7 to 11 levels (e.g., app developer, webmaster)
	manager	publisher account manager: 18 to 26 levels (numeric)
target	Y	if there is a conversion: 2 levels (no, yes)

3.1.2 *Data Preprocessing*

We compare five transformations to handle the nominal inputs: raw (R), categorical or one-hot coding (C), Inverse Document Frequency IDF (I), categorical pruned (CP) and IDF pruned (IP). The categorical coding was computed using only training data. When necessary, training transformation variables (e.g., IDF numeric value for a level) were stored, such that test data could be encoded using the same transform. Moreover, a special "new" category was set to match any new levels present in test data and that could not be known at training time. The transformations work as follows:

- R – uses the original numeric value of the data ("new" is encoded as 0).
- C – assumes a categorical attribute for methods that can handle directly such attributes (e.g., tree based or RF) or one-hot encoding for numeric based methods (e.g., LR). In this second case, the **R** tool transforms each attribute into $L-1$ binary inputs, where L is the number of levels (including the special "new" value).
- CPF – proposed variant used to reduce the input memory requirements. It first ranks the L levels according to their frequency in the data. Only the most frequent F levels are used and all other levels (including "new") are merged into the special "other" category.
- I – coding adopts the transform (Campos et al., 2016): $I(x) = ln(n/f_x)$, where n is the number of instances and f_x is the frequency of attribute x. The transformed $I(x)$ values range from near 0 (most frequent level) to a I_{max} (less frequent).
- IPF – proposed procedure that works by selecting the most frequent F levels, grouping other levels except "new" into an "other" category, and then applying the $I(x)$ function to the $F+1$ levels. In both I and IP procedures, the "new" level is transformed into the least frequent numeric value (I_{max}).

We explore one normal and four balancing methods, which were applied only to training data. Thus, the test sets are kept with their original unbalanced target distributions. The data transformation methods include (Batista et al., 2004): none (N), undersampling (U), oversampling (O), both (B) and SMOTE (S). The N method uses the original data. The U, O and B methods work by sampling the data according to:

- U – uses a sample of Y_{yes} negative examples;
- O – uses a sample (with repetition) of Y_{no} positive examples;
- B – the minority classes are oversampled and the majority classes are undersampled (according to the default **R ROSE** package) (Lunardon et al., 2014); and

- S – creates new synthetic positive examples within the neighborhood input space of the positive labels and undersamples the negative examples, as implemented in the **R DMwR** package (*Torgo*, 2010).

3.1.3 *Classification Methods*

The comparison includes three offline and three online classifiers. The offline algorithms include: Logistic Regression model (LR), Random Forests model (RF) and XGBoost model (XG). LR is a popular linear model for CVR prediction. Both RF and XG are based on decision tree ensembles. RF was proposed in 2001 (*Breiman*, 2001) and it combines the responses of a large number of decision trees. In *Du et al.* (2016) it provided the best CVR prediction results, although it required much more computation than LR. More recently, the scalable XG gradient boosting algorithm was proposed in 2016 (*Chen and Guestrin*, 2016), winning several classification challenges and requiring less computational effort than RF. Regarding the Online Learning algorithms, these include Ozaboost (OB), Decision Stump (DS) and Random Hoeffding Tree (RH). OB is an online boosting ensemble version of the *AdaBoost.M.1* algorithm, DS is based on one-level decision trees and RH uses incremental decision trees (*Bifet et al.*, 2010). All algorithms were implemented in the R tool (*R Core Team*, 2016), using the packages **rminer** (*Cortez*, 2010), for Offline Learning, and **RMOA** (*Bifet et al.*, 2010), for Online Learning. To reduce the bias towards a given algorithm and perform a fair comparison, we executed all algorithms with their default parameters.

3.1.4 *Evaluation*

We adopted the robust rolling window validation (*Tashman*, 2000), which simulates a real classifier usage through time. In the first iteration, the learning model is fit to a training window with the W oldest examples, and then predicts H ahead predictions. Next, the training set is updated by discarding the oldest H records and adding H more recent ones. A new model is fit, producing H new predictions, and so on. In total, this produces $U = D_L - (W + H)$ classifier updates (training and test iterations), where D_L is the data length (number of examples). In this work, after consulting *OLAmobile* experts, we opted to use the realistic values of $W = 50,000$ and $H = 3,000$. The predictive performance is measured using test data and the Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC) curve (*Fawcett*, 2006). Often, the quality of the AUC values is interpreted as: 50% – performance of a random classifier; 60% - reasonable; 70% – good; 80% – very good; 90% – excellent; and 100% – perfect. We also record the computational effort (in seconds) for each rolling window iteration.

The experimental design includes two stages. First, we conduct a large number of preliminary experiments using the oldest collected datasets (duration of 30 minutes), with all preprocessing and classifier combinations. Then, the selected first stage combinations are tested over a larger number

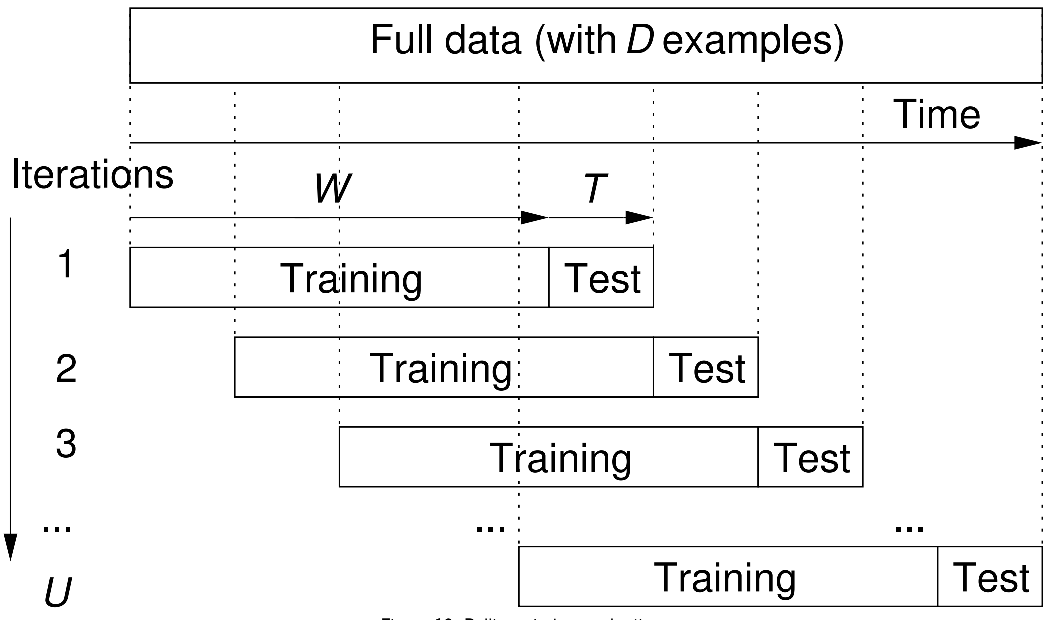

Figure 10: Rolling window evaluation.

of unseen datasets. The goal is to measure the performance of the selected combinations on unseen data. To aggregate all execution results (e.g., AUC values of the U rolling window test iterations), we compute the median value, instead of average values, since this statistic is less sensitive to outliers. All median values are estimated by the Wilcoxon non parametric statistic (*Hollander et al.*, 2013). The same Wilcoxon test is used to check if paired median differences are statistically significant.

3.1.5 *Results*

First Phase

This phase uses only the 30 minutes data. For each factor of analysis (e.g., CP10), there are E executed experiments (e.g., different classifiers). Some experiments produce computational errors (e.g., lack of memory), resulting in R rolling window results ($R \leq E$). For each rolling window execution, we compute the Wilcoxon median result over all U iterations. Then, we compute the Wilcoxon median over all R executions. For the pruned encodings (CP and IP), we tested $F \in \{10, 20, 30\}$. Table 4 presents the median results (AUC and computational effort) when fixing a particular F value, resulting in $E = 5$ (training setups) × 6 (classifiers) = 30 experiments per level and data variant. In Table 4, the number of execution results (R) is shown in brackets and aggregated for both data variants. In a few cases, the $F = 20$ and $F = 30$ levels led to an execution error, resulting in $R = 59$ (and not 60). Since $F = 10$ did not lead to execution errors and provided the best AUC and computational effort overall results, we opted to select this value.

Table 5 presents the overall first phase results for each encoding method. All executions were successful, except for the categorical (C) encoding, confirming that the C transformation is problematic

Table 4: Results for distinct CPF and IPF levels (median values, best values in **bold**)

Traffic	**Variant** (results)	AUC (in %)			Effort (in s)		
BEST		CP10	CP20	CP30	CP10	CP20	CP30
	collected	**84.52**	75.57	72.03	**53.03**	68.26	68.58
	realistic	**82.98**	72.01	72.8	**63.13**	80.64	78.96
	(R)	(60)	(59)	(59)	(60)	(59)	(59)
		IP10	IP20	IP30	IP10	IP20	IP30
	collected	84.49	83.95	**84.61**	**62.93**	63.57	63.82
	realistic	**85.15**	84.04	84.34	82.93	**83.03**	83.34
	(R)	(60)	(60)	(60)	(60)	(60)	(60)
TEST		CP10	CP20	CP30	CP10	CP20	CP30
	collected	**64.30**	63.00	57.85	**42.6**	51.51	58.54
	realistic	**69.47**	66.22	63.78	**52.05**	74.27	74.15
	(R)	(60)	(59)	(59)	(60)	(59)	(59)
		IP10	IP20	IP30	IP10	IP20	IP30
	collected	61.07	59.5	**62.80**	50.06	**49.77**	49.88
	realistic	67.48	66.01	**68.37**	61.71	**61.67**	62.67
	(R)	(60)	(60)	(60)	(60)	(60)	(60)

for high cardinality attributes. The last row presents the overall **median** values, computed over the four data setups. The C encoding also produced the worst AUC values. Considering both the predictive AUC performance (e.g., best overall **median** value of 76.4) and computation effort, we opted to select CP10 as the encoding method for the second stage experiments. The results for a fixed balanced training

Table 5: Results for the categorical encodings (median values, best values in **bold**)

		AUC (in %)					Effort (in s)				
Traffic	**Variant**	CP10	IP10	R	C	I	CP10	IP10	R	C	I
BEST	collected	84.52	84.49	**86.22**	50.54	84.66	**53.03**	62.93	54.79	66.61	61.42
	realistic	82.98	85.15	**85.80**	49.80	85.36	**53.13**	82.93	57.99	77.04	63.14
	(R)	(60)	(60)	(60)	(30)	(60)	(60)	(60)	(60)	(30)	(60)
TEST	collected	**64.30**	61.07	59.80	49.96	60.65	42.64	50.06	**40.58**	55.81	46.53
	realistic	**69.47**	67.48	64.61	51.14	65.17	52.05	61.71	**47.85**	59.21	52.65
	(R)	(60)	(60)	(60)	(30)	(60)	(60)	(60)	(60)	(30)	(60)
	median	**76.4**	75.3	74.5	50.4	74.0	52.3	62.6	**50.3**	64.7	55.9

are shown in Table 6. In this table, $R = 54$ since 6 C encodings produced computational errors. The table includes the **median** value for each traffic type. For the BEST traffic events, the no balancing step (N) achieved the best AUC results. Balancing the training data seems more useful for the TEST traffic, which makes sense, since it presents a lower ratio of sales. Considering both the AUC and computational effort, for the second phase we selected the N training mode for BEST and S for TEST. The last analyzed factor is the learning algorithm (Table 7). The number of executed experiments was $E = 5$ (encodings) × 5 (training setups) × 2 (variants) = 50 experiments. Three learning algorithms produced computational errors for the C encoding, resulting in a smaller $R = 40$. RF achieved the best overall result, followed by XB, OB and LR (for BEST) and followed by LR, OB and XB (for TEST).

Table 6: Results for the training setups (median values, best values in **bold**)

		AUC (in %)					Effort (in s)				
Traffic	**Variant**	N	B	U	O	S	N	B	U	O	S
	collected	**84.13**	80.45	82.98	80.16	82.07	77.15	70.57	80.18	101.80	**25.13**
BEST	realistic	**83.89**	83.31	83.22	83.09	82.25	69.54	70.72	**62.31**	97.78	82.89
	(R)	(54)	(54)	(54)	(54)	(54)	(54)	(54)	(54)	(54)	(54)
	median	**84.01**	81.88	83.03	81.63	82.16	73.34	70.64	71.25	99.79	**54.01**
	collected	57.57	63.45	52.82	63.73	**64.55**	61.60	64.58	53.69	86.28	**15.47**
TEST	realistic	65.45	66.03	62.85	**66.17**	63.48	61.37	60.65	58.70	84.32	**47.67**
	(R)	(54)	(54)	(54)	(54)	(54)	(54)	(54)	(54)	(54)	(54)
	median	61.51	64.74	57.84	**64.95**	64.02	61.48	62.61	56.20	85.30	**37.57**

OB provided the best Online Learning AUC results. Yet, under the adopted rolling window scheme, the computational effort is still higher than LR and XB, being only comparable to RF. For the second phase, we selected three methods: RF (best AUC results), LR (second best TEST results), and XB (fastest method, second best BEST results).

Table 7: Results for the learning methods (median values, best values in **bold**)

		AUC (in %)						Effort (in s)					
Traffic	**Variant**	RF	LR	XB	OB	DS	RH	RF	LR	XB	OB	DS	RH
	collect.	**91.76**	83.95	89.99	87.81	68.10	67.34	90.26	16.21	**7.44**	116.78	95.13	94.52
BEST	realistic	**89.75**	83.78	88.10	87.83	79.24	67.56	100.61	17.93	**9.66**	127.44	106.43	90.73
	(R)	(40)	(40)	(50)	(40)	(50)	(50)	(40)	(40)	(50)	(40)	(50)	(50)
	median	**90.75**	83.87	89.04	87.82	73.67	67.45	95.44	17.07	**8.55**	122.11	100.78	92.62
	collect.	**74.81**	63.38	60.47	64.87	46.67	55.84	83.76	14.90	**8.15**	87.56	78.64	68.92
TEST	realistic	**76.63**	73.06	65.58	68.16	45.74	59.31	97.14	16.44	**8.13**	89.41	80.20	73.28
	(R)	(40)	(40)	(50)	(40)	(50)	(50)	(40)	(40)	(50)	(40)	(50)	(50)
	median	**75.72**	68.22	63.02	66.52	46.2	57.58	90.45	15.67	**8.14**	88.48	79.42	71.10

Second Phase

We tested the combinations that were selected in the first phase (CP10 encoding; N training for BEST and S balancing for TEST; LR, XB and RF) on the unseen datasets (1 hour, 1 day and 1 week). This resulted in $E = 2$ (traffic type) × 2 (data variants) × 3 (durations) = 12 rolling window executions per classifier. There were no computational errors ($R = 12$). The results are shown in Table 8 in terms of median values for all U rolling window iterations for each classifier and data. In terms of AUC, RF is

Table 8: Second phase results (median values, best values in **bold**)

			AUC (in %)			Effort (in s)		
Traffic	**Variant**	**Duration (U)**	**LR**	**XG**	**RF**	**LR**	**XG**	**RF**
BEST	collected	1 hour (62)	72.6	**73.5**[a]	64.9	14.3	**0.6**	58.8
		1 day (99)	79.4	75.0	**83.8**[b]	19.1	**0.6**	55.8
		1 week (70)	72.4	76.0	**80.3**[b]	13.2	**0.5**	75.6
		median	74.3	74.9	**77.3**	15.2	**0.6**	62.3
	realistic	1 hour (61)	70.3	**71.3**[a]	64.5	13.1	**0.5**	55.5
		1 day (60)	71.7	**72.2**[a]	58.8	15.3	**0.6**	61.8
		1 week (44)	69.2	**71.2**[a]	58.0	12.0	**0.5**	53.1
		median	70.4	**71.5**	60.0	13.4	**0.5**	56.5
TEST	collected	1 hour (24)	**68.9**	64.5	68.8	**86.4**	88.4	127.1
		1 day (55)	73.4	68.1	**77.2**[c]	0.4	0.3	1.8
		1 week(24)	67.9	64.1	**68.4**	102.7	93.3	147.0
		median	68.9	65.3	**70.8**	69.0	**67.6**	100.8
	realistic	1 hour (16)	**61.3**	56.0	57.2	17.5	**3.4**	86.0
		1 day (55)	**71.0**	63.9	70.8	9.5	**1.9**	43.0
		1 week (16)	**61.2**	58.3	56.7	18.0	**3.7**	90.5
		median	**63.7**	59.1	60.5	15.6	**3.1**	76.4

a – XG is statistically significant when compared with RF but not LR
b – RF is statistically significant when compared with RF and LR
c – RF is statistically significant when compared with XG but not LR

the best model for the collected setups (BEST and TEST), XG is the best option for BEST and realistic, and LR produces best results for TEST and realistic. The quality of the obtained AUC values can be valued as good (around 70%) or very good (around 80%) for BEST collected; good for BEST realistic and TEST collected (around 70%); and reasonable (around 60%) for TEST realistic. The combination of TEST and realistic is a particularly relevant setup, since the marketing company does not have any information about campaign success in TEST traffic and uses a random user ad matching, which is equivalent to an AUC of 50%. Thus, the proposed LR model has a business value. XG is the fastest method, followed by LR, while RF requires a substantial computation. DSP platforms have real-time requirements, which should be lower than 10 ms for matching users to ads. While we did not use optimized infrastructure and code, several of the XB and LR data-driven models do follow the real-time constrains, even when constantly updating the training model. For instance, for TEST data, LR needs an average of $15.6/3000$=5 ms to issue a prediction, while XG requires a shorter time of 1 ms.

3.1.6 *Conclusions*

In this set of experiments, we studied the user Conversion Rate (CVR) prediction using big data from a global mobile marketing company. Since the company data center receives big data, with millions of ad clicks and thousands of sales per hour, we designed a stream processing engine to collect sample data from the company data center into our computational system. Several datasets with distinct duration times were collected and for two main traffic types: BEST and TEST. Then, we perform an extensive set of CVR prediction tests, under a two stage experimental design and using robust rolling window validation. The first stage explored five categorical transformations, five balanced training setups and six machine learning methods, which were applied to the oldest collected datasets. In the second phase, the best data-driven combinations were then tested on a larger set of unseen datasets.

Interesting predictive performances were achieved, ranging from reasonable (AUC of 61.2% for Logistic Regression and TEST traffic) to very good (AUC of 83.8% for Random Forest and BEST traffic). Thus, there is a potential value for an improved CVR user prediction in the analyzed mobile market. In particular, we achieved an AUC higher than 60% for the realistic TEST scenario, related with new ad campaigns. This is quite valuable for the analyzed DSP, since it currently employs a random user ad matching method for this traffic data type, which corresponds to an AUC of 50%.

Moreover, while we did not use a powerful computational infrastructure or an optimized code, several of the XGboost and Logistic Regression machine learning algorithms did produce real-time results (e.g., $<$10 ms) when performing several training and testing iterations (e.g., U=56).

3.2 Using Deep Learning for Mobile Marketing User Conversion Prediction

Mobile performance marketing is a growing industry due to the massive adoption of smartphones and tablets. In this research, we explored Deep **Multilayer Perceptrons (MLP)** to predict mobile **Conversion Rate (CVR)** when users are redirected to an ad campaign (i.e., if there will be a sale). We analyzed recent real-world big data provided by a global mobile marketing company. Using a realistic rolling window validation, we conducted several experiments with several datasets (two sampling and two data traffic modes), in which we measured both the predictive binary classification performance and the computational effort. The modeling experiments included: two data preprocessing methods, the popular one-hot encoding and a proposed **Percentage Categorical Pruning (PCP)**; and two MLP learning modes, offline (reset) and online (reuse). Overall, competitive classification results were achieved by the **PCP** transform and the two MLP learning modes, producing real-time predictions and comparing favorably against Logistic Regression.

3.2.1 *Stream Engine and Collected Data*

The data was collected during a two week period, starting at 30th May of 2018, using the stream engine presented in Section 3.1.1. The stream engine setup values and number of collected events are shown in Table 9.

We upgraded the stream engine (presented in Section 3.1.1, Figure 9), with an adaptive control request scheme. This adaptive scheme works by increasing or decreasing the number of events to be retrieved in the next query, such that on average we collect the desired NR or NS records per query.

The last column values denote the number of "no sale" (Y_{no}) and "sale" (Y_{yes}) events that were collected. Since we work with sampled data, the sales ratio $R_Y = Y_{\text{yes}}/(Y_{\text{no}} + Y_{\text{yes}})$ is much higher than the expected real DSP ratio. This issue is handled by creating a realistic dataset, in which we randomly undersample (*Batista et al.*, 2004) the number of sales (Y'_{yes}) in order to obtain a more realistic ratio ($R_{Y'}$) of 1.5% for BEST and 0.5% for TEST traffic. Thus, for each traffic mode there are two datasets: collected and realistic.

Table 9: Summary of the stream engine setup and mobile marketing datasets

Traffic	NR	NS	SR	SS	K	Y_{no}	Y_{yes}	Y'_{yes}	R_Y	$R_{Y'}$
TEST	100	100	300	10	2	290,279	29,596	1,600	9.3%	0.5%
BEST	100	100	300	10	2	328,028	156,637	4,847	32.2%	1.5%

Due to technological limitations and privacy concerns, the number of useful features is quite limited in this domain and corresponds to the eleven attributes shown in Table 10. The attributes are related with different entities (column **context**): users, advertisers and publishers. All data attributes are categorical. Some features present a high cardinality, such as the ad campaign and user city.

3.2.2 *Data Preprocessing*

We compare two categorical preprocessing techniques to handle the nominal inputs: One-Hot (1H) encoding and **PCP**. 1H is a popular transform that encodes each attribute into L binary inputs, where L is the number of levels (*Du et al.*, 2016; *Zhang et al.*, 2016). For example, a three categorical feature can be encoded as (1,0,0), (0,1,0) or (0,0,1). In this work, a special "Others" category is added to the set of L levels before executing the one-hot transform. The goal is to encode unseen levels that appear in test data and that could not be known at training time (as shown in Table 13).

The one-hot encoding is popular since it can be applied to any nominal feature. Yet, it highly increases the input space size when there is a high number of levels. Given that most high cardinality features are

Table 10: Data attributes (features)

Context	**Attribute**	**Description** (a – TEST traffic, b – BEST traffic)
user	country	user country: 198^b or 225^a levels (e.g., Russia, Spain, Brazil)
	city	user city: 10690^b or 13423^a levels (e.g., Lisbon, Paris)
	region	region of the country: 23^{ab} levels (e.g., Asia, Europe)
	browser	browser name: 14^{ab} levels (e.g., Chrome, Safari)
	operator	mobile carrier or WiFi: 404^b or 448^a levels (e.g., Vodafone)
advertiser	vertical	ad type: 4^a or 5^b levels (e.g., video, mainstream, dating)
	campaign	ad product identification: 1389^b or 1741^a categorical levels
	special	smart link or special offer: 1018^b or 1101^b levels
publisher	account	publisher type: 8^b or 9^a levels (e.g., app developer, webmaster)
	manager	publisher account manager: 19^b or 34^a categorical levels
target	Y	if there is a conversion: 2 levels (no, yes)

very sparse in this domain, we propose the novel **PCP** transform. It works by first sorting the feature levels according to their frequency in the training data. Then, the least frequent levels (summing up to a threshold percentage of P) are merged into a single category denoted as "Others". Similarly to the one-hot transform, this category is also used to represent unseen levels in test data. Finally, the one-hot encoding is applied using the reduced set of levels, which includes the most frequent levels and the "Others" label.

The goal of the **PCP** transform is to substantially reduce the input memory and processing requirements while keeping the most relevant levels. Fig. 11 exemplifies the effect of this transform by presenting the 1,000 highest frequency values (decreasing order) for the user city attribute for the TEST traffic data (which contains a total of 10,690 levels). For this attribute and when $P = 10\%$, **PCP** selects only the most frequent 688 levels (dashed vertical line in Fig. 11), merging the other 10,002 infrequent levels into the "Others" label. This results in 689 binary inputs, which is much less than the 10,690 binary inputs required by the standard one-hot transform (reduction of $\frac{10690-689}{10690} = 94\%$).

3.2.3 *Multilayer Perceptrons*

In this work, we handle the user mobile marketing CVR prediction as a binary classification task by adopting a MLP, also known as **Deep Feedforward Neural Network (DFFN)** (Goodfellow et al., 2016). The MLP is a fully connected network with several hidden layers. Each node in one layer contains only direct weighted connections to the nodes of the next layer. Let $(L_0, L_1, ..., L_H, 1)$ denote a vector

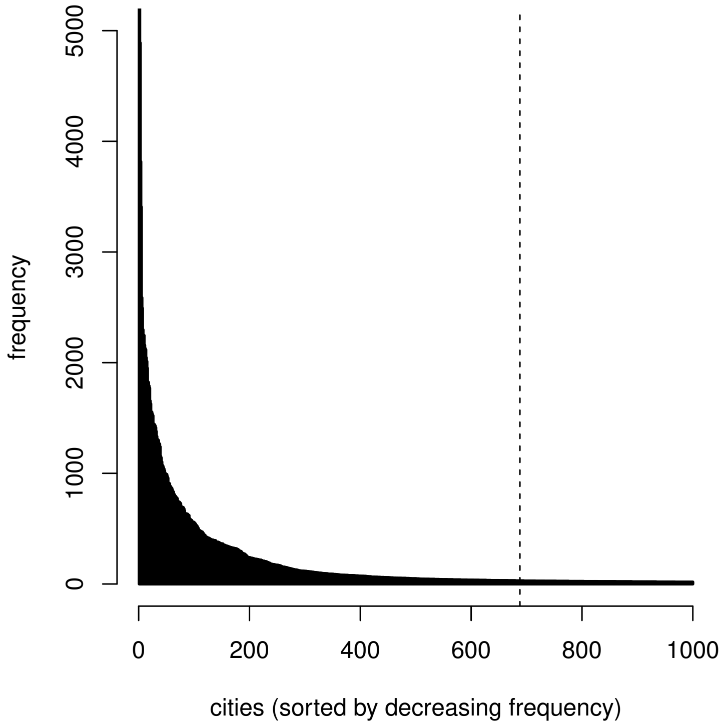

Figure 11: Example of user city most frequent values for TEST traffic (x-axis shows only the 1,000 most frequent levels of a total of 10,690 categorical values; y-axis presents the frequency of the x-axis levels)

with the layer sizes, where $L_0 = I$ is the input layer size, H is the number of hidden layers and there is one output node. Each MLP node computes:

$$\begin{aligned} z_{k:m,j} &= w_{m:0,j} + \sum_{i \in \{1,...,L_{m-1}\}} w_{m:i,j} a_{k:m-1,i} \\ a_{k:m,j} &= f(z_{k:m,j}) \end{aligned} \tag{10}$$

where $z_{k:m,j}$ denotes the weighted sum of the j-th node of layer m and for a given node k, $w_{m:i,j}$ the weight connection from node i (of previous layer) to node j (of current layer m), $a_{m,j}$ the activation value for the same node and f the activation function. The $w_{m:0,j}$ weights are known as bias. For the input layer ($m = 0$), $a_{k:0,i} = x_{k,i}$ (the input values).

The design of the MLP structure often involves heuristics and trial-and-error experiments. Since the number of inputs is quite large in this domain and there is just one output, in this work we opted to use a triangular shape MLP, in which each subsequent layer size is smaller: $L_0 > L_1 > ... > L_H > 1$. In preliminary experiments, conducted using older mobile marketing data, collected before May 2018, we compared some combinations of triangular MLP structures (e.g., $L_1 \in \{1000, 1024\}$, $L_2 \in \{500, 512\}$) and dropout rates within {0.2,0.3,0.4,0.5} in some of the hidden layers. After these trial-and-error experiments, we selected a fixed MLP structure with $H = 9$ hidden layers: $(I, 1024, 512, 256, 128, 64, 32, 16, 8, 2, 1)$. In all hidden layers ($m \in \{1, ..., 9\}$) we used the popular ReLU activation function, due to its fast training and good convergence properties (LeCun et al., 2015). As for the output node, we used the logistic function, which allows to compute class probabilities ($\in [0,1]$). These functions are computed as:

$$\begin{array}{ll} \text{ReLU} & max(0, z_{k:m,j}) \\ \text{Logistic} & \frac{1}{1+\exp^{-z_{k:m,j}}} \end{array} \quad (11)$$

During the training phase, we used the AdaDelta gradient function (*Zeiler*, 2012), which is an efficient stochastic gradient decent method. We used two approaches to avoid overfitting: dropout and earlystopping. Dropout randomly ignores weighted connections and it was applied on hidden layers $m = 4$ and $m = 6$ with the values 0.5 and 0.2. Earlystopping was perfomed by monitoring the binary cross-entropy loss function on a validation set (with 30% of the training data). The training algorithm was stopped when the validation error increases or after a maximum of 100 epochs.

In the marketing domain, several works assume a static offline environment, in which one model is trained once and then used to predict the user CTR or CVR values (*Zhang et al.*, 2014; *Du et al.*, 2016; *Zhang et al.*, 2016; *Lu et al.*, 2017). However, when working with stream data, the learning models should be more dynamic, assuming an Online Learning scenario, with several model updates (*Ramírez-Gallego et al.*, 2017). In this experiment, we test this effect by setting two MLP learning modes: reset and reuse. These two modes are used with the rolling window validation scheme described in Section 3.2.4.

The reset mode follows a standard Offline Learning procedure where a new **MLP** model, fully initialized with random weights, is trained when new training data is available. The reuse approach uses an Online Learning strategy, where the weights of the previously trained **MLP** are first stored. As previously explained, new training data often contains unseen input levels (e.g., new campaigns). If that is the case, after preprocessing the data (with one-hot or **PCP**), the corresponding new input nodes and weights (using random initialization) are added to the previously trained **MLP** model. Fig. 12 exemplifies this procedure by showing the first two **MLP** layers when one input level (node I from layer $m = 0$) is added under the reuse mode. Next, the whole new **MLP** is retrained, using the AdaDelta gradient function.

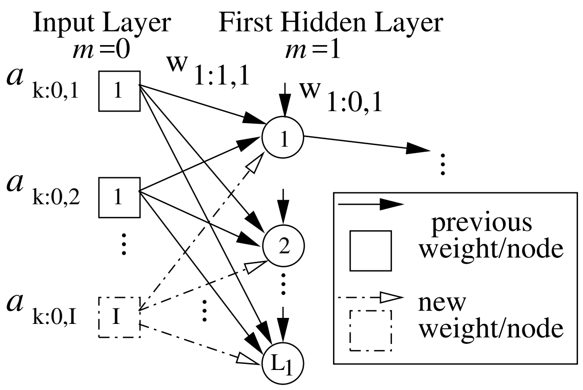

Figure 12: Example of a new input level added during the reuse mode

3.2.4 *Evaluation*

The learning models are evaluated using the robust rolling window validation (*Tashman*, 2000), which simulates a real classifier usage through time, with several training and test updates as presented in Fig. 10. In the first iteration ($u = 1$), the classifier model is adjusted to a training window with the W oldest examples, and then predicts T test ahead predictions. Next, in the second iteration ($u = 2$), the training set is updated by discarding the oldest T records and adding T more recent ones. A new model is fit, producing T new predictions, and so on. In total, this produces $U = D - (W + T)$ model updates (training and test iterations), where D is the data length (number of examples). After consulting OLAmobile experts, we opted to use the realistic values of $W = 100,000$ and $H = 5,000$. For our datasets, this leads to the following number of model updates: $U = 42$ – collected TEST traffic; $U = 75$ – collected BEST traffic; $U = 38$ realistic TEST traffic; and $U = 46$ realistic BEST traffic.

The predictive classification performance is measured using the receiver operating characteristic (ROC) curve (*Fawcett*, 2006) on the rolling window test data. If a classifier outputs a class probability p_k (for example k), then the class can be interpreted as positive if $p_k > K$, where K is a fixed decision threshold, otherwise it is considered negative. The curve shows the performance of a two class 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 the area under the curve ($AUC = \int_0^1 ROCdK$). It is common to interpret the quality of the AUC values as: 0.5 – equal to a random classifier; 0.6 – reasonable; 0.7 – good; 0.8 – very good; 0.9 – excellent; and 1 – perfect.

For each rolling window iteration, we store the AUC value on test data and also the computational effort (in seconds). To aggregate all $u \in \{1, ..., U\}$ execution results, we compute the median value, which is less sensitive to outliers when compared with the average. We also adopt the Wilcoxon non parametric statistic (*Hollander et al.*, 2013) to check if paired median differences are statistically significant.

3.2.5 *Results*

All experiments were implemented in Python, using the popular Keras library (*Chollet*, 2017). For the **PCP** transform, we assume a cutting threshold of $P = 10\%$ (shown in Fig. 11) for all input attributes. The classification quality metrics (e.g., AUC) were executed using the rminer library of the R tool (*Cortez*, 2016). We used a multi-core Linux server (Intel Xeon 1.70GHz with 56 cores), where each classification experiment was executed in a distinct core.

In the first set of experiments, we compare the proposed **MLP** with 2 other models implemented in Keras: the classical **Logistic Regression (LR)** and a **Convolutional Neural Network (CNN)**. The CNN was configured based on the Keras library sequential model guidelines, namely the usage of six convolution layers with 3 kernels, of sizes (128, 128, 64, 64, 32, 32) and ReLU activation functions, followed by a 1D maxpooling layer, another 16 layer convolutional layer, a flatten layer and then a **MLP** with hidden layers of (512, 256 – with dropout of 0.5, 32, 16 – dropout of 0.2, 8, 1 – with the logistic function). The CNN layers act as a filter that reduces the input space. Since the CNN training required a substantial computational effort, we considered only the first $u \in \{1, 2, 3\}$ iterations of the rolling window procedure and the collected BEST traffic data.

Table 11 shows the first comparison results. For the neural network models (**MLP** and CNN), the last column details the number of training epochs (using early stopping). The results confirm that the CNN model requires a much higher training time while providing often a worst predictive classification performance (in terms of AUC) when compared with **MLP**. Thus, in the additional paper experiments we only compare **MLP** with LR.

Table 11: Comparison of **MLP** (reset mode) with CNN and LR models for collected BEST data.

Preprocess	**Iteration**	**Model**	**AUC**	**Effort (s)**	**Epochs**
1H	$u = 1$	MLP	0.88	962	22
	$u = 2$		0.90	688	14
	$u = 3$		0.91	720	15
	median		0.90	720	15
	$u = 1$	CNN	0.30	32901	3
	$u = 2$		0.90	137490	13
	$u = 3$		0.50	32206	3
	median		0.50	32901	3
	$u = 1$	LR	0.77	13	–
	$u = 2$		0.80	12	–
	$u = 3$		0.81	13	–
	median		0.80	13	–
PCP	$u = 1$	MLP	0.89	133	17
	$u = 2$		0.89	105	14
	$u = 3$		0.90	123	14
	median		0.89	123	14
	$u = 1$	CNN	0.50	15758	8
	$u = 2$		0.50	11834	6
	$u = 3$		0.90	17383	9
	median		0.50	15758	8
	$u = 1$	LR	0.76	13	–
	$u = 2$		0.78	12	–
	$u = 3$		0.79	13	–
	median		0.78	13	–

In the second set of experiments, we executed the full rolling window procedure over all datasets (assuming different preprocessing, sampled data and traffic types), aiming to perform a robust comparison of the distinct categorical transforms and learning models. Table 12 shows the obtained results, aggregated by the median over all rolling window executions. As expected, LR is the fastest method, requiring a median training time around a few seconds for each rolling window iteration. However, the LR predictive classification performance (in terms of AUC values) is always inferior, presenting statistical significant differences when performing a pairwise Wilcoxon test with **MLP** reset and **MLP** reuse. The AUC differences range from 6 (1H preprocessing, collected BEST data) to 35 (**PCP** preprocessing, realistic TEST data) percentage points. In particular, the LR classification capability is highly affected when using the unbalanced training data (realistic data), obtaining AUC values that are close to the random classifier performance (AUC of 0.50). In contrast, the Deep Learning methods obtain a high quality predictive discrimination, with AUC values ranging from very good (AUC of 0.85) to excellent (AUC of 0.92). We note that the more realistic scenario is highly relevant for the mobile marketing domain. And the results confirm that the Deep Learning models are robust for this data type, showing their class discrimination robustness when learning from unbalanced training data.

Table 12: Full rolling window results (best value per dataset in **bold**).

Preprocess	Data	Traffic	Model	AUC	Effort (s)
1H	collected	TEST	MLP reset	**0.92**^{ac}	485
			MLP reuse	0.87^{c}	367
			LR	0.72	**14**
		BEST	MLP reset	**0.89**^{ac}	399
			MLP reuse	0.84^{c}	369
			LR	0.78	**14**
	realistic	TEST	MLP reset	0.86^{c}	104
			MLP reuse	**0.88**^{bc}	109
			LR	0.52	**12**
		BEST	MLP reset	0.86^{c}	101
			MLP reuse	**0.87**^{c}	103
			LR	0.53	**12**
PCP	collected	TEST	MLP reset	**0.92**^{ac}	157
			MLP reuse	0.87^{c}	76
			LR	0.67	**3**
		BEST	MLP reset	**0.89**^{ac}	149
			MLP reuse	0.84^{c}	83
			LR	0.77	**2**
	realistic	TEST	MLP reset	0.85^{c}	40
			MLP reuse	**0.87**^{bc}	45
			LR	0.50	**2**
		BEST	MLP reset	**0.85**^{c}	41
			MLP reuse	**0.85**^{c}	47
			LR	0.50	**2**

a - statistically significant when compared with MLP reuse.
b - statistically significant when compared with MLP reset.
c - statistically significant when compared with LR.

When analyzing the categorical transform methods, Table 12 confirms that **PCP** reduces substantially the training time. As shown in Table 13, the **PCP** diminishes the number of **MLP** binary (0 or 1) inputs by a factor of 7.9 (8319/1051) or 9.4 (7785/830), in terms of the median levels that are kept from the previous rolling window iteration. Similar reduction level factors are produced for the new levels: 8.0 (183/23) and 9.1 (181/20). Having less levels translates into a smaller computational effort in Table 12. For example, the reduction assumes a factor of 4.7 (485/104) for **MLP** reset and collected TEST data and a factor of 9.0 (750/83) for **MLP** reuse and collected BEST data. As for the predictive performance, the results are either similar (e.g., **MLP** reset – AUC of 0.92 for collected TEST) or very close (e.g., 1 percentage point difference for **MLP** reset realistic TEST traffic), without statistical significance. These results highlight the importance of the **PCP** transform in this marketing domain.

Table 13: Number of median input levels used during the rolling window procedure for the collected data.

Traffic	Preprocess	Kept levels	New levels	Total
TEST	1H	8319	183	8503
	PCP	1051	23	1074
BEST	1H	7785	181	7966
	PCP	830	20	850

Regarding the offline (reset) and online (reuse) learning modes, in four cases the offline mode requires less computational effort: 1H collected TEST (reduction factor of 1.32) and BEST (reduction factor of 1.08), and **PCP** collected TEST (reduction factor of 2.06) and BEST (reduction factor of 1.80). When the computational demand increases, it is very slight (e.g., increase factor of 1.05 for 1H collected BEST). This behavior is justified by the preprocessing time to sort and handle new input attribute levels (as shown in Table 13). When analyzing the classification predictive discrimination, there is not clear gain of one learning mode when compared with the other. Reset obtains the best AUC results in four cases (with statistical significance), reuse is the best option in other three cases (two with statistical significance) and there is a tie in the last comparison experiment of Table 12.

As an example to demonstrate the quality of the predictions, we present additional results when using the **PCP** transform and realistic BEST traffic data. The evolution of the AUC test values for all $u \in \{1, ..., 46\}$ rolling window iterations is plotted in Fig. 13. The plot confirms a stable discrimination performance through time of the proposed **MLP** reuse and reset models. The second rolling window iteration ($u = 2$) predictions are further detailed in Fig. 14, which presents the individual ROC curves. In this figure, the two **MLP** models present a class discrimination that is much better than LR. Both MLPs present similar ROC curves, with the **MLP** reuse mode slightly obtaining a better sensitivity result (higher TPR for higher FPR values, top right region of the curve) when compared with **MLP** reset for this example.

Using the same rolling window iteration predictions ($u = 2$), we perform an additional quality analysis by computing the accumulated LIFT curves (Fig. 15), which are very popular in the marketing domain (*Moro et al.*, 2015a). The curve is built by first sorting, using a decreasing order, the test examples according to the respective classifier probabilities for a positive result (e.g., sale). The ordered samples are ploted in the x-axis and the accumulated percentage of actual sales, achieved up to a

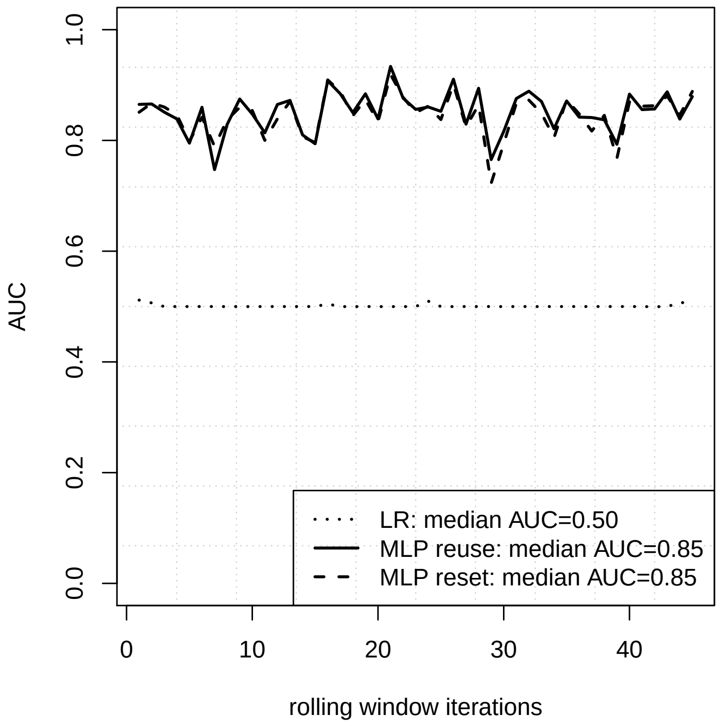

Figure 13: Evolution of AUC values (y-axis) for all rolling window iterations (x-axis) when using the **PCP** transform and realistic BEST traffic data

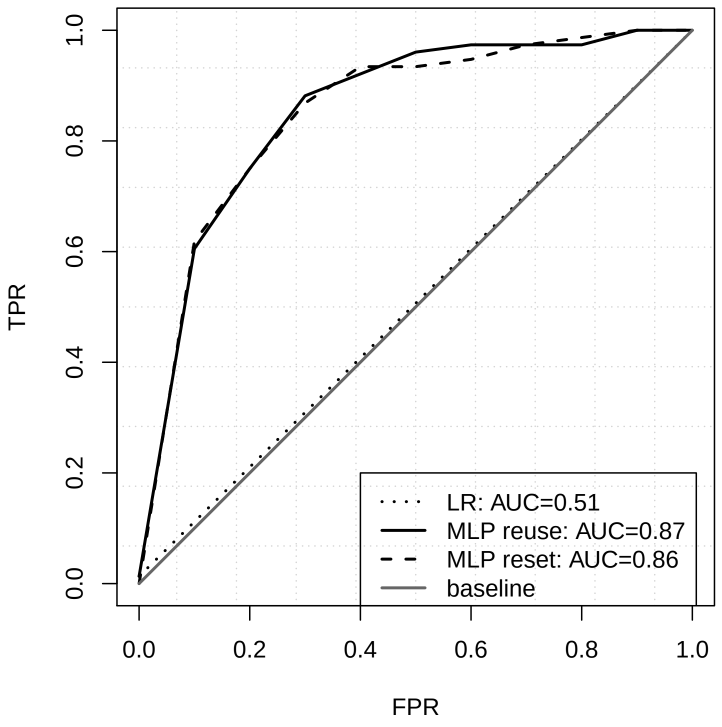

Figure 14: ROC curves for PCP realistic BEST dataset and second rolling window iteration ($u = 2$; x-axis denotes the False Positive Rate – FPR, while y-axis represents the True Positive Rate – TPR)

particular ordered sample percentage, is shown in the y-axis. The higher the accumulated LIFT area (ALIFT), the better is the model predictive performance to select the users willing to buy a product. For example, Fig. 15 shows that if only 11% of all user clicks were redirected to an ad, as selected by the

MLP reset or reuse predictive models, then 60% of the sales would be obtained. Overall, considering all sample percentages, Fig. 15 reveals that both **MLP** models present a high quality LIFT, confirming that a high discrimination performance (AUC of 0.86 or 0.87) is correlated with a high acummulated LIFT result (ALIFT of 0.86).

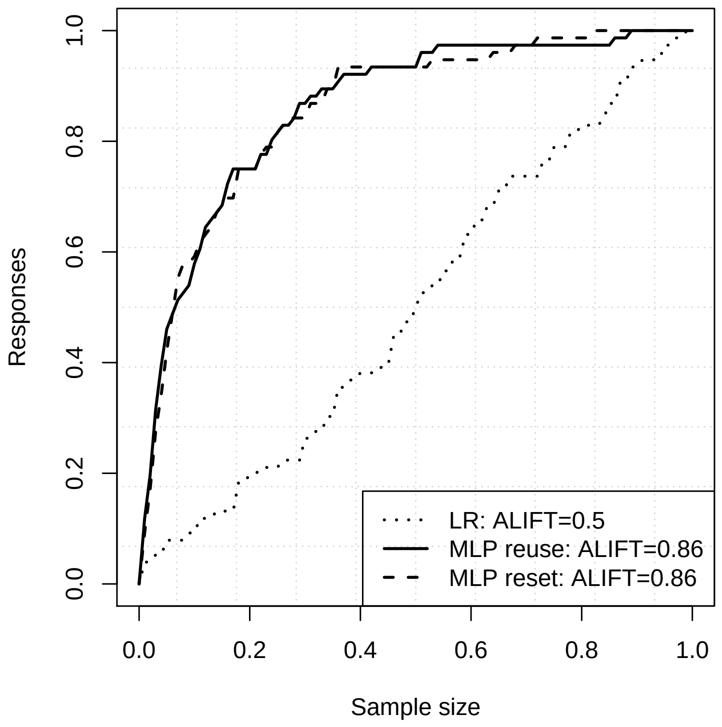

Figure 15: Accumulated LIFT curves for PCP realistic BEST dataset and second rolling window iteration ($u = 2$; x-axis denotes the percentage of test examples, while y-axis represents the accumulated percentage of actual sales)

3.2.6 *Conclusions*

In this research, we have worked with recent DSP big data, from a worldwide acting company, *OLAmobile*. Given that the company works with millions of user clicks per hour, we first designed a stream processing engine to collect sampled data from the data center. Using a two week collection period, we have created several datasets, assuming two main types of traffic (TEST and BEST) and two sampling modes (collected, with all retrieved events, and realistic, with lower sale ratios). Then, we conducted several classification ("no sale","sale") experiments using two categorical transform methods (one-hot – 1H and Percentage Categorical Pruning – PCP) and two **Deep Learning Multilayer Perceptron (MLP)** modes (reset – offline and reuse – online).

The experiments were conducted using a realistic rolling window validation procedure, in which both the predictive performance, measured using Receiver Operating Characteristic (ROC) curves, and computational effort were analyzed. Interesting predictive user CVR performances were achieved, with area under the ROC curve (AUC) values higher than 85% for the **MLP** models, corresponding to a very good or excellent discrimination. The obtained results are competitive when compared with other models, such as Logistic Regression (LR). In effect, the tested MLPs maintain their high discrimination power even when they are trained with highly unbalanced data (corresponding to the realistic datasets with percentages of sales of: 0.5% – TEST and 1.5% – BEST). Moreover, the **PCP** transform substantially reduces the computational effort (e.g., reduction by a factor of 4.7 or 9.0) while keeping the same predictive discrimination level. As for the **MLP** learning mode comparison, the reuse mode performs a significant computational effort reduction when compared with the reset mode in half of the conducted experiments. Similar predictive discrimination performances were achieved by the two **MLP** approaches, with the offline reset mode providing the best AUC values in four cases and the online reuse mode obtaining the highest AUC values in three other cases.

From the point of view of the analyzed DSP market, the most interesting experiments are related with the realistic sampled data and **PCP** transform. For BEST traffic, both **MLP** reset and reuse achieve an AUC level of 85%. As for the TEST traffic, **MLP** reuse obtains an AUC of 87%. These results are valuable. For example, the mobile marketing company currently employs a random ad campaign selection for the TEST traffic operation mode, which corresponds to an AUC level of 50%. This suggests that the proposed **MLP** model predictions, with a 37 percentage point improvement in terms of AUC, could substantially boost the DSP user ad matching performance. Moreover, although we did not use a powerful computational infrastructure (e.g., GPU processor) or an optimized code, the **MLP** models are capable of producing real-time predictions. For example, under the analyzed rolling window scheme, the **MLP** reuse TEST traffic model full training and testing requires a median of 45 seconds, producing a total of 5000 predictions, which results in an average of 9 ms per prediction, a value that is lower than the 10 ms that are required by this marketing domain.

3.3 Using Deep Learning for Ordinal Classification of Mobile Marketing User Conversion

In this work, we explore Deep Multilayer Perceptrons (MLP) to perform an ordinal classification of mobile marketing conversion rate (CVR), for predicting the value of product sales when an user clicks on an ad. As a case study, we consider big data provided by a global mobile marketing company. Several experiments were held, considering a rolling window validation, different datasets, learning methods and performance measures. Overall, competitive results were achieved by an online deep learning model, which is capable of producing real-time predictions.

3.3.1 *Data Collected*

We collected the data from a worldwide marketing company (OLAmobile). The DSP generates two main events: redirects – the user ad clicks; and sales – when there is a conversion. All redirects and sales are stored at the DSP data center, being associated with a timestamp when they arrive. The DSP managed by the company generates millions of redirects and thousands of sales per hour.

We had access to a secure web service that allowed us to retrieve NR redirects and NS sales from the data center. Our computing server, an Intel Xeon 1.70GHz wth 56 cores and 2TB of disk space, is limited when compared with the data center and thus we work with sampled data. The data was collected during a two week period, starting at 30th May of 2019, via a stream engine that uses K computing cores to continuously retrieve redirects and sales, 'sleeping" every SR and SS seconds as presented in Matos et al. (Table 14).

Table 14: Summary of the collected DSP datasets

Traffic	NR	NS	SR	SS	K	Y_{no}	Y_{yes}	Y'_{yes}	R_Y	$R_{Y'}$
TEST	100	100	300	10	2	290,279	29,596	15,393	9.3%	5.04%
BEST	100	100	300	10	2	328,028	156,637	17,389	32.2%	5.03%

The analyzed DSP contains two traffic modes: TEST – used to measure the performance of new campaigns; and BEST – with 80% of the traffic and including only the best TEST performing ads. The Y_{no} and Y_{yes} columns denote the number of collect "no sale" and "sale" events. Also, the R_Y column represents the sales ratio $R_Y = Y_{\text{yes}}/(Y_{\text{no}} + Y_{\text{yes}})$. Since we worked with sample data, the collected data ratio is higher than the expected real DSP one. To get a more realistic dataset, we randomly undersample (Batista et al., 2004) the number of sales (Y'_{yes}) such that a more realistic ratio ($R_{Target'}$) is obtained, which in this work was fixed to 5% for both BEST and TEST traffic. Thus, for each traffic mode there are two datasets: collected (C) and realistic (R). Table 15 presents a summary of the collected attributes. Most attributes are categorical and some present a high cardinality (e.g., city).

After consulting the DSP experts, we created the ordinal Y target by grouping the value into $k = 5$ classes (Fig. 16). The grouping was achieved by using rounded EUR values after using quantiles over the collected sales values (when >0): "none" – equal to 0; "very low" – > 0 and < 0.03; "low" – $>= 0.03$ and < 0.10; "medium" – $>= 0.10$ and < 0.30; "high" – $>= 0.30$. As shown in Fig. 16, when there is a sale, the ordinal classes are relatively balanced. Also, TEST traffic presents a lower number of sales when compared with BEST traffic.

3.3.2 *Data Preprocessing and Ordinal Approaches*

Since several attributes are sparse and present a high cardinality (Table 3), we applied the Percentage Categorical Pruned (PCP) transformation to all input attributes (Matos et al., 2019a). The transformation works by merging the least 10% frequent levels in the training data into a "others" category. Then, the resulting values are preprocessed using the known one-hot coding, which assigns one binary input per

Table 15: Summary of the DSP data attributes

Context	**Attribute**	**Description** (a – TEST traffic, b – BEST traffic)
user	country	user country: 198^b or 225^a levels (e.g., Russia, Spain, Brazil)
	city	user city: 10690^a or 13423^b levels (e.g., Lisbon, Paris)
	region	region of the country: 23^{ab} levels (e.g., Asia, Europe)
	browser	browser name: 14^{ab} levels (e.g., Chrome, Safari)
	operator	mobile carrier or WiFi: 404^b or 448^a levels (e.g., Vodafone)
advertiser	vertical	ad type: 4^a or 5^b levels (e.g., video, mainstream, dating)
	campaign	ad product identification: 1389^b or 1741^a categorical levels
	special	smart link or special offer: 1018^b or 1101^b levels
publisher	account	publisher type: 8^b or 9^a levels (e.g, app developer, webmaster)
	manager	publisher account manager: 19^b or 34^a categorical levels
sale	value	value of the conversion in EUR: numeric (e.g., 0.00, 0.01, 69.34)

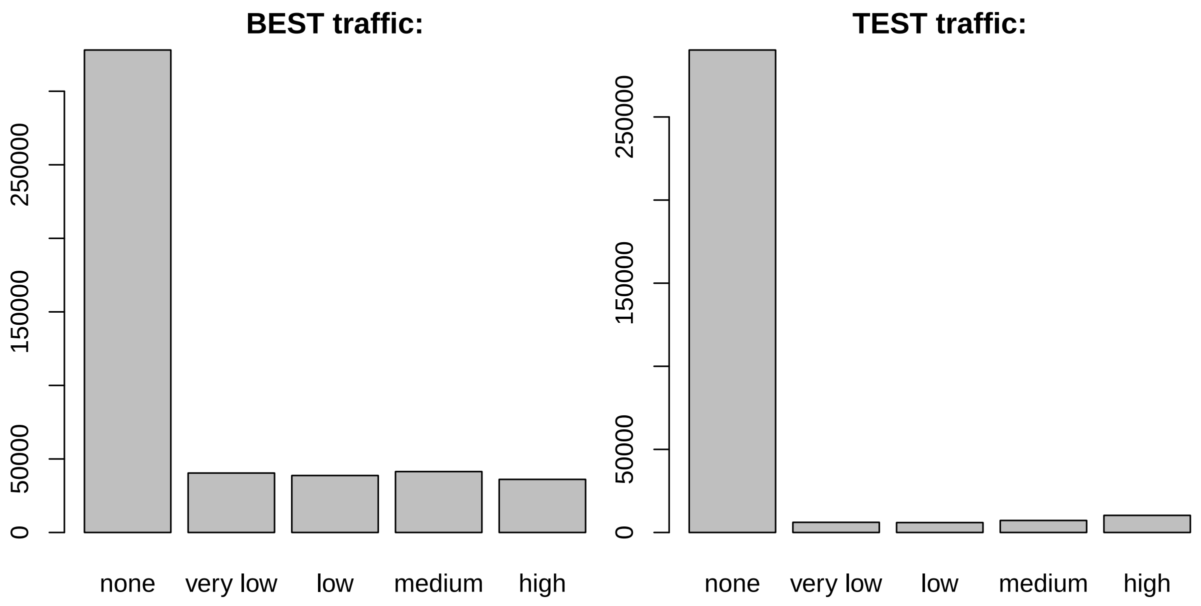

Figure 16: Histograms for the ordinal sale classes for collected BEST and TEST datasets

level. In *Matos et al.* (2019a), the PCP transform allowed to substantially reduce the input memory (e.g., reduction of 94% for the city attribute) and processing requirements.

For the mobile marketing data, the number of conversions (sales) is typically much lower than the number of ad clicks (redirects). While the target CVR data is unbalanced, in *Matos et al.* (2019a) we found that the binary deep learning MLP classifier was capable of high quality predictions even without any training data balancing method. However, when approaching the ordinal task and in particular for the realistic datasets (which present the lowest 5% conversion rate), the training data becomes extremely unbalanced. Thus, we opted to balance the realistic training data by using the SMOTE method (*Batista* et al., 2004), which creates new synthetic examples for the minority classes. This balancing method was only applied to training data and thus the test sets were kept with the original target values.

In this paper, we apply three approaches for the ordinal classification: Multi-class Classification (MC), regression and the k-1 Ordinal Approach (OA). The first approach discards the ordering and performs a simpler 5-class classification task. The second approach transforms each ordinal class into a numeric score y, where $y \in [0, 1, 2, 3, 4]$ (R1) or $y \in [0, 1, 2, 4, 8]$ (R2). The first regression scale (R1) uses equal spaced values, while the second one (R2) assigns larger distances to the highest sales ("medium" and "high"), in an attempt to favor such classes due to the minimization of squared errors. In both scales, the ordinal class is associated with the nearest scale value to the prediction (e.g., in R2 a prediction of 3.1 is assumed as the "medium" class). The third approach transforms the ordinal target, with the classes $V1 < V2 < ... < V_k$, into k-1 binary tasks (Frank and Hall, 2001). Each classifier learns the probability of $P(Y > V_k)$, $k \in \{1, ..., k-1\}$. Then:

$$P(V_1) = 1 - P(Y > V_1) \tag{12}$$

$$P(V_i) = \max(0,\ P(Y > V_{i-1}) - P(Y > V_i)\ ), 1 < i < k \tag{13}$$

$$P(V_k) = P(Y > V_{k-1}) \tag{14}$$

The classifiers are independent and in a few cases we experimentally found it could occur that $P(Y > V_i) > P(Y > V_{i-1})$. Thus, we added the $\max(0, ...)$ function in Eq. (13) to avoid the computation of negative probabilities.

3.3.3 *Deep Learning Methods*

In previous work, a very competitive deep learning MLP model was proposed for binary CVR prediction, outperforming a convolution neural network and a logistic regression algorithm (Matos et al., 2019a). We adapt the same MLP, also known as Deep Feedforward Neural Network (DFFN) (Goodfellow et al., 2016). Let $(L_0, L_1, ..., L_H, L_O)$ denote a vector with the layer sizes with $m \in \{1, ..., H\}$ hidden layers, where $L_0 = I$ is the input layer size (I is the total number of input levels after the PCP transform) and L_O is number of output nodes. The adopted model consists of a trapezoidal shaped MLP, with $H = 8$ hidden layers of decreasing size: $(I, 1024, 512, 256, 128, 64, 32, 16, 8, L_O)$. In all hidden layers ($\{1, ..., 8\}$) we used the popular ReLU activation function, due to its fast training and good convergence properties. For multi-class classification, the output layer contains L_O=5 nodes and the softmax function is used to output class probabilities $P(V_k) \in [0, 1]$. For the regression models, only $L_O = 1$ linear output node is used. Finally, for the $k-1$ ordinal classification, one logistic output node ($L_O = 1$) is used, trained such that $P(Y > V_k) \in [0, 1]$.

During the training phase, we used the AdaDelta gradient function (Goodfellow et al., 2016), which is based on a stochastic gradient descent method. Following our previous work (Matos et al., 2019a), we used two approaches to avoid overfitting: dropout, which randomly ignores neural weights (dropout values of 0.5 and 0.2 for hidden layers $m = 4$ and $m = 6$); and early stopping, which stops the training when the validation error does not improve 1% within 3 epoch runs of after a maximum of 100 epochs.

Since we work with stream data, the learning models should be dynamic, assuming a continuous learning through time. We compare two MLP learning modes (proposed in Matos et al.) for ordinal classification: reset – offline mode, when new training data is available, the whole neural weights are randomly set; and reuse – online learning, any new training starts with the previous fitted MLP weights (from previous rolling window training) and only the new input node (due to appearance of new input levels) to the first hidden layer connections are randomly set.

3.3.4 *Evaluation*

The learning models are evaluated using a robust rolling window validation (*Oliveira et al.*, 2017), which simulates a classifier usage through time, with multiple training and test updates. In the first iteration ($u = 1$), the model is adjusted to a training window with the W oldest examples and then predicts T test predictions. In the next iteration ($u = 2$) , the training set is updated by discarding the oldest T records and adding T more recent ones. A new model is fit, producing T new predictions, and so on. In total, this produces $U = D-(W+T)$ model updates (training and test iterations), where D is the data length (number of examples). After consulting OLAmobile experts, we opted to use the realistic values of $W = 100,000$ and $H = 5,000$, which results in the following model updates: $U = 43$ – collected TEST traffic; $U = 76$ – collected BEST traffic; $U = 41$ realistic TEST traffic; and $U = 49$ realistic BEST traffic.

For each rolling window iteration, we collect the test data measures and the computational effort (in seconds). We compute the F1-score, F1$_k$ for each class V_k, which considers both precision and recall (*Witten et al.*, 2017). The global measure is obtained by using the Macro-averaging F1-score (MF1), which weights equally the F1-score for each class. We note that the ordinal classes are unbalanced, and thus other F-score averaging measures, such as micro or weight averaging, would favor mostly models that classify well the no conversion "none" class. As a secondary global measure, we adopt the Mean Absolute Error for Ordinal Classification (MAEO) (*Sousa et al.*, 2013), which computes how far (using absolute errors) are the predictions from the target (e.g., the error is 2 if the prediction is "low" and the target is "high"). This measure is often used in ordinal classification but, similarly to the micro and weight averaging F1-score measures, it tends to produce low values when the classifier is more biased to correctly predict the "none" class. The rolling window results are averaged over all U iterations and the Wilcoxon test is used to check if paired MF1 differences are significant (*Hollander et al.*, 2013).

3.3.5 *Results*

The experiments were coded in Python using the Keras library *Chollet* (2017). We tested two types of datasets (C and R), two types of traffic (BEST and TEST), three ordinal methods (MC, R1/R2, OA), two MLP learning modes (reset, reuse) and a LR baseline model. Table 16 shows the obtained average rolling window classification results. Overall, the best learning algorithm is MLP reuse, which tends to produce the highest MF1 values, often with statistical significance when compared with MLP reset or LR model. When compared with LR, MLP reuse presents an MF1 improvement that ranges from 7 to 15 percentage points. The reuse learning always requires less computational effort when compared with the reset mode. As for the ordinal methods, the multi-class (MC) and $k-1$ OA achieve the best overall MLP reuse results, with slight F1-score differences. In general, the regression ordinal scales (R1 and R2) produce worst F1-score results. Only in two cases (R BEST and TEST), R2 obtained the best F1-score for the "high" conversion class (V_5). The $k-1$ OA method requires more computation than the MC approach. Yet, we note that in this work we used one processing core for each model, thus the OA effort could by substantially reduced if $k-1$ cores were used to fit each of its individual binary models.

3.3.6 *Conclusions*

In this research, we used big data from a mobile marketing company. The goal was to predict the type of Conversion Rate (CVR) when an user clicks an ad, set in terms of five ordinal classes. Using a realistic rolling window validation, we compared three main ordinal methods (multi-class – MC, regression – R1/R2 and $k-1$ ordinal approach – OA) using two deep learning approaches (offline – MLP reset; and online – MLP reuse) and a logistic regression (LR) model.

The best results were achieved by the MLP reuse model and the MC and OA approaches. As an added bonus, this model is capable of performing these predictions in real-time. For instance, the 5,000 predictions for the C BEST MC setup require 42 s, which results in an average of 8 ms per prediction. Interesting results were achieved for the collected (C) datasets, with most F1-scores above 50%, macro F1-scores of 64% and 54%, as well as a low MAEO error (lower than 0.5). As for the realistic (R) datasets (with lower amount of conversion cases), while low MAOE errors where obtained (e.g., lower than 0.30), the individual F1-scores are lower when compared with the collected datasets, resulting in an average macro F1-score of 38% and 45%.

Considering all obtained results, we recommend the MC MLP reuse model, which requires the training of just one classifier and is capable of real-time predictions. This model is potentially capable of providing value to the analyzed marketing company, since it currently does not have any method to estimate the value level of a conversion. In particular, for TEST traffic the company uses a random selection of ads, which produces much lower macro F1-scores. For instance, for the realistic TEST dataset, the random class assignment results in an average macro F1-score that is around 10%.

Table 16: Classification results (best values in **bold**; best models highlighted in gray).

Data	Traffic	Meth.	Model	Global		F1-score per class					Effort
				MAEO	**MF1**	**F1$_1$**	**F1$_2$**	**F1$_3$**	**F1$_4$**	**F1$_5$**	(s)
C	BEST (U=76)	MC	MLP reset	0.48	0.57	0.87	0.49	0.35	0.54	0.58	61
			MLP reuse	**0.45**	**0.64**[a]	**0.88**	**0.56**	**0.48**	**0.61**	0.66	42
			LR	0.51	0.57	0.86	0.51	0.38	0.51	0.59	**35**
		R1	MLP reset	0.63	0.22	0.85	0.22	0.03	0.01	0.00	46
			MLP reuse	0.61	0.24	0.84	0.30	0.04	0.00	0.00	44
		R2	MLP reset	0.77	0.28	0.83	0.00	0.00	0.31	0.24	49
			MLP reuse	1.02	0.28	0.77	0.00	0.00	0.26	0.39	47
		OA	MLP reset	0.47	0.59	0.87	0.51	0.43	0.54	0.61	117
			MLP reuse	**0.45**	**0.64**[b]	**0.88**	0.55	**0.48**	**0.61**	**0.67**	94
			LR	0.51	0.54	0.86	0.47	0.35	0.49	0.54	39
	TEST (U=43)	MC	MLP reset	0.25	0.19	0.95	0.00	0.00	0.00	0.00	46
			MLP reuse	0.19	0.51	**0.96**	**0.36**	0.22	0.45	0.55	45
			LR	0.21	0.43	0.96	0.26	0.14	0.36	0.45	**31**
		R1	MLP reset	0.22	0.22	**0.96**	0.08	0.02	0.01	0.01	51
			MLP reuse	0.22	0.23	**0.96**	0.15	0.03	0.00	0.00	45
		R2	MLP reset	0.28	0.30	0.95	0.00	0.00	0.24	0.33	50
			MLP reuse	0.27	0.31	0.95	0.00	0.00	0.24	0.34	47
		OA	MLP reset	0.21	0.31	**0.96**	0.10	0.09	0.20	0.19	118
			MLP reuse	**0.18**	**0.54**[b]	**0.96**	0.34	**0.31**	**0.50**	**0.57**	109
			LR	0.21	0.42	**0.96**	0.24	0.13	0.34	0.41	39
R	BEST (U=49)	MC	MLP reset	0.37	0.37	0.91	0.20	**0.19**	0.24	0.29	100
			MLP reuse	0.31	**0.38**[c]	**0.93**	**0.21**	**0.19**	**0.26**	0.30	**84**
			LR	0.93	0.26	0.76	0.13	0.12	0.13	0.15	120
		R1	MLP reset	0.32	0.21	0.88	0.10	0.07	0.01	0.00	106
			MLP reuse	**0.30**	0.21	0.89	0.10	0.07	0.00	0.00	87
		R2	MLP reset	0.68	0.25	0.87	0.00	0.00	0.08	0.28	104
			MLP reuse	0.61	0.26	0.89	0.00	0.00	0.08	**0.33**	88
		OA	MLP reset	0.34	0.37	0.92	0.19	**0.19**	0.24	0.30	278
			MLP reuse	0.37	0.36	0.92	0.20	**0.19**	0.22	0.25	219
			LR	0.89	0.25	0.73	0.10	0.10	0.14	0.19	109
	TEST (U=41)	MC	MLP reset	0.23	0.44	**0.96**	**0.27**	**0.26**	0.33	0.40	130
			MLP reuse	0.22	**0.45**[c]	**0.96**	**0.27**	**0.26**	**0.35**	0.41	99
			LR	0.69	0.30	0.83	0.13	0.11	0.17	0.23	156
		R1	MLP reset	0.19	0.25	0.95	0.18	0.10	0.03	0.00	132
			MLP reuse	**0.18**	0.23	0.95	0.17	0.04	0.00	0.00	103
		R2	MLP reset	0.30	0.28	0.95	0.00	0.00	0.14	0.33	130
			MLP reuse	0.29	0.31	0.95	0.00	0.00	0.15	**0.46**	**97**
		OA	MLP reset	0.21	0.44	**0.96**	0.24	0.24	0.34	0.42	303
			MLP reuse	0.21	0.44	**0.96**	0.25	**0.26**	0.33	0.42	240
			LR	0.65	0.29	0.81	0.09	0.11	0.18	0.28	123

a – Statistically significant when compared with MC reset and LR.

b – Statistically significant when compared with OC reset and LR.

c – Statistically significant when compared with MC LR.

3.4 A Deep Learning Based Decision Support System for Mobile Performance Marketing

In this work, we propose an IDSS to automatically select mobile marketing campaigns for users. The IDSS is based on a computationally efficient mobile user conversion prediction model that assumes a novel PCP categorical preprocessing and an online deep MLP model. Such model outperformed an offline MLP and a LR, obtaining a high quality class discrimination when applied to sampled (85% to 92%) and complete (90% to 94%) data. Moreover, we designed two strategies (A — best campaign selection; and B — random selection among the top candidate campaigns) to build the IDSS, in which the predictive deep learning model is used to perform a real-time selection of advertisement campaigns for mobile users. Using recently collected big data (with millions of redirect events) from a worldwide mobile marketing company, we performed a realistic IDSS evaluation that considered three criteria: response time, potential profit and advertisers' diversity. Overall, competitive results were achieved by the IDSS B strategy when compared with the current marketing company ad assignment method.

When contrasted with the state-of-the-art works (Section 2.14.1 and Section 2.14.2), we note that our previous work (Section 3.2) was the first study that proposed deep learning for mobile performance marketing user CVR prediction. It also addressed other relevant issues:

- Most CTR or CVR prediction studies tend only to consider prediction classification measures and not the computational effort (*Du et al.*, 2016; *Zhang et al.*, 2016; *Wen et al.*, 2020) For instance, the deep learning models proposed in *Zhang et al.* (2016) are more complex than the LR method, although the classification only improved very slightly (e.g., 0.1 percentage points). In contrast, we assessed the obtained deep learning models in terms of both the predictive performance and the computational effort.

- Several studies only consider static offline learning scenarios by using a single train and test holdout validation scheme (*Zhang et al.*, 2014; *Du et al.*, 2016; *Zhang et al.*, 2016; *Wen et al.*, 2020) In contrast, we used a realistic rolling window scheme, which considered several training and test iterations through time, to evaluate the all data-driven models. We have also proposed a reuse learning mode that is more suited for dynamic time changes, since it learns from previously trained neural networks.

- Most CTR or CVR works use the popular one-hot encoding to handle categorical inputs (*Zhang et al.*, 2014; *Du et al.*, 2016; *Zhang et al.*, 2016; *Wen et al.*, 2020) which heavily increases the computational effort for high cardinality input features. Instead of using the one-hot encoding, we proposed a new PCP transform, which substantially reduces the memory and computational requirements of the predictive models.

However, the previous work of Section 3.2 contained two main limitations. First, it analyzed only sampled data, collected in 2018 by using a developed data stream engine. The sampled data, termed collected data, contained a higher sales ratio than what would be expected to occur with with the real DSP. This issue was handled by creating another dataset, called realistic, with an undersample of the sales events. Nevertheless, such undersampling is synthetic and does not accurately reflect the true data distribution values (see Table 17). Second, and similarly to other related works, it did not analyze how the predictive models could be used in a real DSP environment to select mobile advertisements for users. This paper extends our previous work by handling both these limitations. Rather than working with sampled data, we had access to larger and complete datasets, collected in the year of 2019.

3.4.1 *Mobile Marketing Data*

We worked with data from OLAmobile, which is a worldwide mobile performance company that is responsible for its own DSP. The analyzed DSP records two main event types: redirect, each time a user clicks a dynamic ad; and sales, when there is a conversion. Also, it works under two traffic modes: TEST – used to measure the performance of new campaigns; and BEST – with only the best TEST performing ads and that corresponds to most of the DSP data. We designed a data stream engine that allowed us to collect sampled data during a two week period, starting at 30th May of 2018. More recently, we had a direct access to the complete from the company via a datacenter available on an Amazon server. The complete data includes all events received by the company between November 15th, 2019, and November 18th, 2019, for the BEST and TEST traffic modes. Table 17 summarizes the main characteristics of the DSP datasets. While related with a smaller time period (four days), the complete data contains a much higher number of examples, with millions of redirects and thousands of sales. As previously discussed, the sampled realistic overall sales to redirects ratio is close but not identical to the real DSP ratio. For instance, in Matos et al. (2019a) we used a realistic synthetic ratio of 1.5% for the BEST traffic, but the real ratio for the complete BEST data is 2.8%.

Table 17: Summary of the mobile marketing data.

Fetch method	**Year**	**Period**	**Name**	**Mode**	**Nº no conversions**		**Nº conversions**	
sampled	2018	2 weeks	collected	TEST	290,279	(90.7%)	29,599	(9.3%)
				BEST	328,028	(67.7%)	156,637	(32.3%)
			realistic	TEST	290,279	(99.5%)	1,600	(0.5%)
				BEST	328,028	(98.5%)	4,847	(1.5%)
complete	2019	4 days	complete	TEST	2,283,725	(99.0%)	22,769	(1.0%)
				BEST	4,076,375	(97.2%)	112,532	(2.8%)

Due to technological limitations and privacy concerns, the number of useful features is quite limited in this domain and corresponds to the attributes shown in Table 18. The attributes are related to different entities (column **context**): users, advertisers and publishers. All data attributes are categorical. Some features present a high cardinality (e.g., ad campaign). It should be further noted that DSPs tend to evolve through time, resulting in several data features changes. The city attribute was available in the DSP data in 2018 and thus was used in *Matos et al.* (2019a,b). However, this attribute was discarded in 2019 and thus it is not included in the data that was collected for this study.

Table 18: Summary of the DSP data attributes.

Context	**Attribute**	**Description** (a – TEST traffic, b – BEST traffic)	**Sampled**	**Complete**
user	country	user country: 198 to 225 levels (e.g., Russia, Spain, Brazil)	✓	✓
	city	user city: up to 13423 levels (e.g., Lisbon, Paris)	✓	✗
	region	region of the country: 23 levels (e.g., Asia, Europe)	✓	✓
	browser	browser name: 14 levels (e.g., Chrome, Safari)	✓	✓
	operator	mobile carrier or WiFi: up to 448 levels (e.g., Vodafone)	✓	✓
advertiser	vertical	ad type: 4 to 12 levels (e.g., video, mainstream, dating)	✓	✓
	campaign	ad product identification: up to 1741 levels	✓	✓
	special	smart link or special offer: up to 1101 levels	✓	✓
publisher	account	publisher type: 8 to 10 levels (e.g., app developer, webmaster)	✓	✓
	manager	publisher account manager: 10 to 34 categorical levels	✓	✓
target	Y	if there is a conversion: 2 levels (no, yes)	✓	✓

3.4.2 *Data Preprocessing*

Most CVR works adopt the one-hot encoding to transform categorical attributes into numeric ones (*Du et al.*, 2016; *Zhang et al.*, 2016). This transform assumes one binary input per categorical level. For instance, the three levels {“a”,“b”,“c”} can result in the following one-hot encoding: “a” → (1,0,0), “b” → (0,1,0) and “c” → (0,0,1). However, as explained in Section 3.2.2, the one-hot encoding creates a vast amount of inputs when the attribute cardinality is high, resulting in more computational effort (in terms of memory and training time) for the machine learning algorithms. In this mobile marketing domain, there are several high cardinality features that are very sparse. Thus, in Matos et al. (2019a) we proposed the use of the Percentage Categorical Pruned (PCP) transform and the effect of this preprocessing method is exemplified in Fig. 11.

3.4.3 *Multilayer Perceptrons*

In this work, we handle the user mobile marketing CVR prediction as a binary classification task by adopting a DFFN (Goodfellow et al., 2016; Matos et al., 2019a), as presented in Section 3.2.3.

Thus, our model has a fixed MLP structure with $H = 9$ hidden layers with the following hidden unit scheme: $(I, 1024, 512, 256, 128, 64, 32, 16, 8, 2, 1)$. In all hidden layers ($m \in \{1, ..., 9\}$) we used the popular ReLU activation function, due to its fast training and good convergence properties (*Goodfellow et al.*, 2016; *Moolayil*, 2019).

During the training phase, we used the AdaDelta gradient function (*Zeiler*, 2012), which is an efficient stochastic gradient decent method (*Okewu et al.*, 2019). We used two approaches to avoid overfitting:

dropout and earlystopping. Dropout randomly ignores weighted connections and it was applied on the hidden layers $m = 4$ and $m = 6$ with the values of 0.5 and 0.2. Earlystopping was performed by monitoring the binary crossentropy loss function on a validation set (with 30% of the training data). The training algorithm was stopped when the validation error increased or after a maximum of 100 epochs.

The reset mode follows a standard offline learning procedure where a new MLP model is fully initialized with random weights when new training data is available. In contrast, the proposed reuse approach assumes an online learning approach. Thus, the weights of the previously trained MLP are first stored. As previously explained, new training data often contains unseen input levels (e.g., new ad campaign).

However, in this work, we devise two main strategies to assign advertisements to users:

A - Using the highest conversion probability (p_i), based on the combination of the advertiser features (vertical, special, campaign), the most probable advertisement is presented to the end-user;

B - performs a random selection within the top $B = 10$ best advertisements (the ones with the highest p_i values).

Strategy **A** will result in higher conversion rates and consequently should increase the expected profit. However, it also should decrease the campaign diversity and consequently always select a smaller range of products and advertisers. By including some randomness in the campaign selection procedure, the second strategy (**B**) will increase the diversity of the selected campaigns, making the DSP less dependent of a small subset of advertisers.

3.4.4 *Evaluation*

The analyzed DSP datasets are related with two different time periods (Table 17): two weeks (sampled) and four days (complete). Therefore, the rolling window procedure assumes two setups. For the collected data, the procedure is controlled by the number of data examples, with $W = 100,000$, $T = S = 5,000$, resulting in U =38 to U =75 model updates (depending on the analyzed {collected, realistic} and {TEST, BEST} combination) as stated in *Matos et al.* (2019a).

As for the complete data, we adjusted the rolling window to be controlled by fixed time periods, which is closer to what would occur in a real DSP environment. Thus, the training window corresponds to all training examples produced during a two-day period, while the predictive model is tested on a hourly basis. This leads to $U = 48$ training and testing model updates.

During the execution of the two main rolling window procedures, we store the computational effort (time elapsed, in seconds). The predictive classification performance was measured using the receiver operating characteristic (ROC) curve (*Fawcett*, 2006) on the rolling window test data. If a classifier outputs a class probability p_k (for example k), then the class can be interpreted as positive if $p_k > K$, where K is a fixed decision threshold, otherwise it is considered negative. The curve shows the performance of a two class classifier across all ($K \in [0, 1]$) values plotting one minus the specificity

(x-axis) versus the sensitivity (y-axis). The area gives the overall discriminatory performance named AUC (area under the curve) which is computed:

$$AUC = \int_0^1 ROC\, dK \tag{15}$$

It is common to interpret the quality of the AUC values as: 0.5 – equal to a random classifier; 0.6 – reasonable; 0.7 – good; 0.8 – very good; 0.9 – excellent; and 1 – perfect.

The second rolling window experiments, using the recent DSP data consisting of four days of all the transactions, are also used to simulate the advertisement selection and compare the proposed **Intelligent Decision Support System (IDSS)** strategies (**A** and **B**) with the OLAmobile Demand-Side Platform (DSP) assignment method (termed **O** strategy). The IDSS strategies use the selected user CVR prediction model to estimate the expected conversion probability p_i for a candidate advertisement a_i. In each iteration of the rolling window, the hourly test data is used to define the set of active advertisements (A, total of n_A distinct values). Moreover, the same test data is used to define the redirect context (user and publisher attributes). We further note that the target output (Y) of the test data corresponds to the **O** strategy selection. The OLAmobile method selects the most profitable campaign for a given user context (the four user attribute values from Table 18). When a user arrives for the first time to the DSP, the selected campaign is the most profitable one, generally the first row of the fixed table. Recurrent visits of the same user, result in moving down on that table, which means moving to the second most profitable campaign, then the third one and so on.

During the advertisement selection experiments, we record the computational effort (in *ms*). One important constraint of the DSP is that a dynamic link needs to be issued within a maximum response time of 10 *ms*. We also measure the conversion probability (p_i), assigned to the selected advertisement (a_i). Finally, we compute two diversity measures: Variety and True Saved Space (TSS).

Let $X =< x_1, x_2, ..., x_T >$ denote a sequence that contains the assigned advertisements for the T test set user redirects, where $x_i \in \{a_1, ..., a_{n_A}\}$ (e.g., $X =< a_1, a_2, a_1, a_3, a_2, ... >$). The Variety is a simple indicator that counts the number of distinct advertisements that are included in X. As for TSS, it is a newly indicator that is based on the information entropy concept proposed by Claude Shannon (*Shannon*, 1948):

$$H(X) = -\sum_{i=1}^{n_A} P(X = a_i) \log_2(P(X = a_i)) \tag{16}$$

where $P(X = a_i) \in [0, 1]$ is computed as the proportion of a_i assignments included in X. The entropy computes the number of bits necessary to represent information and it is often used as a diversity measure. The higher the entropy, the more diverse is the X distribution. However, this entropy depends on the number of possible outcomes. In our case, this corresponds to the number of active advertisements (n_A), which varies every hour during the rolling window simulation. To handle this issue, we propose the TSS measure that computes the percentage of saved space when compared with the equally probable assignment method (the most diverse scenario):

$$TSS = 1 - \frac{H(X)}{\log_2(n_A)} \tag{17}$$

where $H(X)$ is the entropy of the analyzed campaign distribution and $\log_2(n_A)$ is the entropy for the equally probable assignment. The lower the TSS value, the higher will be the diversity of the strategy.

3.4.5 *Results*

All experiments were implemented in Python, using the popular Keras library (*Chollet*, 2017). The categorical processing was executed using a newly implemented Python module, named Categorical Attribute traNsformation Environment (CANE)[2] (*Matos et al.*, 2020). We used a 2.3 GHz Intel Core i9 processor, where each classification experiment was executed in a unique core. Table 19 presents the results that were obtained in the study published in *Matos et al.* (2019a) in terms of predictive accuracy (AUC) and computational effort (including preprocessing, training and testing times, in seconds). Using the sampled datasets, several configurations were compared: two preprocessing methods (1H - one-hot encoding; PCP - Percentage Categorical Pruned); two conversion/redirects ratios (collected and realistic); two traffic modes (TEST and BEST); and three machine learning algorithms (LR - Logistic Regression; MLP reset - a new MLP is trained in each rolling window iteration; and MLP reuse – which retrains the previously used MLP). As explained in *Matos et al.* (2019a), the best overall results were obtained by the PCP MLP reuse model, which is much faster model than 1H while achieving a high quality classification discrimination (in terms of AUC values).

Given the results presented in Table 19, we have selected the PCP MLP reuse algorithm and applied it to the more recent and complete 2019 data. In these experiments, the rolling window assumes fixed time periods, with a two-day training window and an hour testing. The obtained results are shown in Table 20. The complete datasets involve large training sets. On average, 1.5 and 0.9 millions of records are used to train the BEST and TEST traffic predictive models. Nevertheless, the required computational effort is reasonable, requiring around 162 and 107 seconds to process and train two days of complete DSP data. Moreover, an excellent predictive discrimination level was obtained by the reuse deep learning model, obtaining AUC values of 90% (BEST traffic) and 94% (TEST data). We note that the deep learning model was designed in *Matos et al.* (2019a) using sampled data from the year of 2018. When applied to the complete and more recent data (year of 2019), the same deep structure kept the same high quality user CVR discrimination level, confirming that the proposed model is robust to dynamic changes.

The PCP reuse MLP data-driven model was used in the advertisement assignment experiments. The analyzed average number of redirects and active advertisements are shown in Table 20. The

2 Available at: `https://pypi.org/project/cane/`.

Table 19: Results obtained in work (Matos et al., 2019a) (best value per dataset in **bold**).

Preprocess	Data	Traffic	Model	AUC	Effort (s)
1H	collected	TEST	MLP reset	0.90^{ac}	205.45
			MLP reuse	$\mathbf{0.92}^{c}$	152.63
			LR	0.72	**142.31**
		BEST	MLP reset	0.88^{ac}	228.26
			MLP reuse	$\mathbf{0.89}^{c}$	140.96
			LR	0.78	**132.79**
	realistic	TEST	MLP reset	0.76^{c}	**131.49**
			MLP reuse	$\mathbf{0.88}^{bc}$	134.87
			LR	0.51	144.38
		BEST	MLP reset	0.82^{c}	123.04
			MLP reuse	$\mathbf{0.86}^{c}$	127.03
			LR	0.53	**124.82**
PCP	collected	TEST	MLP reset	0.88^{ac}	21.12
			MLP reuse	$\mathbf{0.92}^{c}$	16.93
			LR	0.67	**16.79**
		BEST	MLP reset	$\mathbf{0.88}^{ac}$	21.64
			MLP reuse	$\mathbf{0.88}^{c}$	14.98
			LR	0.77	**13.99**
	realistic	TEST	MLP reset	0.77^{c}	**14.56**
			MLP reuse	$\mathbf{0.85}^{bc}$	14.75
			LR	0.50	15.68
		BEST	MLP reset	0.80^{c}	**12.62**
			MLP reuse	$\mathbf{0.85}^{c}$	12.67
			LR	0.50	12.80

a - statistically significant when compared with MLP reuse.
b - statistically significant when compared with MLP reset.
c - statistically significant when compared with LR.

Table 20: Rolling window results for the PCP MLP reuse model and complete DSP data (average values over all $U = 48$ iterations).

	BEST	**TEST**
Preprocess Effort (s)	50.11	30.06
Training Time (s)	111.62	77.15
Training set size	1,573,742	903,213
Test set size	60,778	34,916
Active campaigns (n_A)	107	146
AUC	0.90	0.94

ad assignment comparison is presented in Table 21. In the table, the Time measure refers to the computational effort (in ms) that is required to process a single redirect and perform an ad assignment. Both IDSS strategies (**A** and **B**) are fast, requiring around 4 ms, which is around half the DSP time limit of 10 ms. The CVR measures the estimated average conversion probability (in %). As expected, the conversion rates are higher for BEST traffic when compared with the TEST data. Overall, the best conversion is provided by the greedy **A** strategy, followed by the second IDSS method (**B**). The CVR differences are statistically significant when compared with the DSP strategy (**O**). The obtained gain is 8 (BEST) and 3 (TEST) percentage points when comparing **A** with **O**, and 5 (BEST) and 2 (TEST) percentage points when comparing **B** with **O**. On the other hand, the diversity measures (Variety and TSS) position the DSP assignment method (**O**) as the more diverse one, followed by the **B** strategy, which is more diverse than **A**.

Table 21: Advertisement assignment comparison (average over all $U = 48$ iterations).

	A		**B**		**O**	
	BEST	TEST	BEST	TEST	BEST	TEST
Time (ms)	3.933	4.183	3.935	4.215	-	-
CVR (%)	12.38	4.47	9.45	3.17	4.06	1.42
Variety	46	29	78	68	440	587
TSS	0.67	0.77	0.45	0.5	0.37	0.32

3.4.6 *Conclusions*

In this research, we have worked with recent DSP big data, from a worldwide acting company, OLAmobile, conducting several computer simulations for campaign selection for the user based on the models created in Matos et al. (2019a). Using sampled data, collected in the year of 2018, and a realistic rolling windows evaluation procedure, we compared: two categorical preprocessing methods, one-hot encoding and Percentage Categorical Pruned (PCP); and three machine learning algorithms, Logistic Regression model (LR) and two deep Multilayer Perceptron (MLP) methods (reset and reuse). Overall, the best results were obtained by the PCP MLP reuse method. This model was selected for further rolling window experiments, conducted using complete and more recent DSP data, from the year of 2019. Besides estimating the predictive performance, the model was adapted to perform advertisement assignments, based on two strategies: **A** - selection of the highest conversion probability ad; and **B** - random selection within the top ten candidate ads. While working with larger training sets, the selected PCP MLP reuse method required a reasonable computational effort (e.g., less than 3 minutes to process two days of data, with millions of records). Moreover, it obtained an excellent user CVR prediction performance (e.g., 90%). Also, when adopted to assign advertisements (via the **A** and **B** strategies), the model achieved real-time responses (much lower than the 10 ms DSP limit) and a global conversion rate that is significantly higher than the current DSP assignment method. We also measured the diversity of the selected advertisements via two criteria, Variety and a proposed True Saved Space (TSS) indicator. Both diversity measures favored strategy **B** when compared with **A**.

The final selection of the best DSP ad assignment method requires setting the right trade-off between conversion and diversity. Higher conversion methods will increase the revenue for the advertisers, publishers and DSP company. On the other hand, less diverse methods will reduce the number of advertisers and sold products, while also increasing the irritation of users (e.g., showing often the same ad). The obtained results were shown to the analyzed DSP company, which provided a very positive feedback and favored strategy **B** as an interesting conversion versus diversity trade-off.

3.5 Summary

This chapter presents, following a chronological order, a set of four research works that resulted in three conference papers and a paper submitted to a journal.

In the first work (*Matos et al.*, 2018), we performed an initial ML comparison study, aiming to explore different preprocessing (e.g., one-hot, IDF, a newly proposed CP method), balancing (e.g., none, SMOTE) and ML algorithms (using both a offline and online learning) for binary user CVR prediction. We also developed an initial data stream collection engine, since our computational server was quite limited (in terms of storage and memory capability) when compared with the DSP full datacenter. The ML approaches were compared by using a two stage experimental design and using robust rolling window validation. Overall, the best predictions were obtained by the offline learning algorithms, namely Logistic Regression model (LR), Random Forests model (RF) and XGBoost model (XG).

The second work (*Matos et al.*, 2019a) explored Deep Learning models, more specifically based on the Deep Feedfoward Network (DFFN). In this research, the data stream collection engine was further improved and more recent datasets were sampled from the DSP data center. The best binary user CVR prediction results were obtained using a PCP categorical encoding and the reuse learning mode Deep Feedfoward Network (DFFN). It was also interesting to note that high quality predictive results (e.g., with AUC>85%) were obtained without using any balanced training technique, showing that the Deep Feedfoward Network (DFFN) model can deal well with unbalanced data.

The third work targeted an ordinal user CVR prediction (*Matos et al.*, 2019b). Using the same data stream collection engine from the previous work, we sampled more recent data from the DSP data center. Then, we explored a novel ordinal classification of mobile user CVR, which assumes five classes (from zero to a high revenue of the conversion): “no sale”, “very low”, “low”, “medium” and “high”. These ordinal classes provide a good proxy to the client Lifetime Value (LTV). Thus, by using such ordinal classifier, a DSP could better select the best ad campaign for a particular user by maximizing the expected conversion value. Overall, the best results were obtained by a multi-class MLP reuse method. While some interesting results were achieved, the F1-scores for some of the infrequent classes were below 50% (Table 16). In effect, the ordinal classification results were shown to the DSP company, which signaled that a better predictive performance was needed prior to the deployment of the ordinal classifiers. Due to limitations related with the duration of the R&D project PROMOS and this PhD thesis, we opted to suspend the ordinal classification research and focus better on the development of the IDSS (executed in the last work).

The fourth and last research (Matos et al.) targeted a realistic simulation of the proposed Intelligent Decision Support System (IDSS), which was based on the binary user CVR prediction model designed in (Matos et al., 2019a) (Deep Feedfoward Network (DFFN) with a PCP categorical encoding and the reuse learning mode). One interesting aspect of this work is that we adopted a complete collection of the DSP data, rather than just working with sampled data. The complete dataset is related with four days of data and it includes around 6 millions of redirect events. We compared two main strategies for selecting ads for users: **A** - best campaign selection; and **B** - random selection within the top best candidate campaigns. When compared with the current ad assignment method adopted by the analyzed DSP (**O**), the two strategies are competitive. It should be noted that the DSP company provided a very positive feedback to the obtained IDSS results (see Chapter 4).

4

CONCLUSIONS

This final chapter outlines the main conclusions from this work. It is divided in two main sections. Firstly, Section 4.1 presents a summary of the overall PhD work and discusses the obtained results and limitations. Then, Section 4.2 presents suggestions of future work.

4.1 Overview and Discussion

The worldwide usage of mobile devices (e.g., smartphones, tablets) has increased the market value of the mobile performance marketing industry. In these mobile markets, monetary compensation only is executed when an ad performs well, leading to a "conversion". These markets are typically implemented by using a **Demand-Side Platform (DSP)** and they involve several types of players:

- **advertisers** who want to sell products or services and that adhere to a DSP with the expectation of increasing their sales;
- **users** who often want a "free" access to the content provided by the publishers;
- **publishers** who want to fund their content or web services (e.g., online news, games) by requiring users to first view an advertisement prior to accessing the content (typically, publishers attract a vast audience of users); and
- **the DSP owner and manager** that acts as a broker, assuring the connection between users and advertisers and the monetization feedback of the conversion revenues.

The DSP performs several real-time operations, including the assignment of ads to dynamic links clicked by the users. This PhD work is inserted within the R&D project PROMOS, which involves the OLAmobile mobile marketing company. When the project started, the company identified a critical aspect to improve the performance of the managed DSP. Currently, the OLAmobile DSP assumes simple statistic rules to assign ads to users, thus the company wanted to research on the design of a **Machine Learning (ML)** algorithm to better assign the ad to the user.

Most related works aim to estimate the **Conversion Rate (CVR)** probability that a specific user will perform a purchase when viewing an ad. As shown in Table 1, there have been several ML approaches to user **Conversion Rate (CVR)** prediction. Most of the works adopt simple categorical to numeric transforms (one-hot encoding) and conduct offline computation experiments, often with lack of realism (e.g., usage of a single train-test holdout split). Moreover, the works focus mostly on predictive performance but not on computational effort, which is relevant in this domain due to the real-time DSP needs. In addition, few studies have approached Deep Learning architectures. Also, none of the surveyed works have used the binary user CVR prediction models to actually select ads and consistently measure the usefulness of such models through time.

The main goal of this work is the design and development of a **Intelligent Decision Support System (IDSS)** capable of providing value to the analyzed DSP. As explained in Section 1.3.2, developing such a IDSS is a non trivial task, because: it involves big data; only a small portion of user clicks result in a conversion; due to privacy and technological limitations, the set of data features is quite reduced; all features are categorical and several of these features often present a high cardinality (e.g., with thousands of levels).

In this PhD work, we explored four research directions, aiming to fill the literature research gaps and also handle the complexity associated with the IDSS for this domain. First, we performed an initial comparison of distinct ML approaches (including several preprocessing approaches, and balanced training methods and ML algorithms) for binary mobile user CVR prediction (*Matos et al.*, 2018). Following the increased impact of **Deep Learning (DL)** on the ML field, we then approached several DL architectures for the same binary CVR prediction task (*Matos et al.*, 2019a). The proposed **Deep Feedfoward Network (DFFN)** with a **Percentage Categorical Pruned (PCP)** encoding and reuse learning obtained the best overall results. Next, we adapted the best DL models to a ordinal (and more informative) user CVR estimation task (*Matos et al.*, 2019b). While the best results were obtained by a multi-class version of the previously proposed DFFN, the predictive results were considered unsatisfactory for a real deployment setting by the considered DSP company. Finally, using the best model obtained in the second published work (*Matos et al.*, 2019a), we designed a IDSS prototype that is capable of assigning ads to users (*Matos et al.*). The developed IDSS included two assignment strategies, which were compared against the standard DSP ad assignment method. The results were shown to the DSP company, which considered the proposed IDSS to provide satisfactory and interesting results.

During the exploration of the four research works, there was a need to work with different data storage and processing technologies. Initially, we worked with a static dump file, provided by the company. Since this data was rather limited, we then asked the company for a more dynamic access to the data. Given that our computed server, acquired for the PROMOS R&D project, had memory and storage limitations, we opted to work with sampled data. Thus, we created a data stream engine, in order to dynamically retrieve data from the company data center. This development raised several technical challenges, such as handling failed responses from the data center and also the need to simultaneously use several processing cores, thus producing a data collection with simultaneous samples at different time rates. Moreover, it was not easy to find the right balance between fetching redirects and sales events, which resulted in an adaptive adjustment of some stream engine parameters, as adopted in (*Matos et al.*, 2019a).

Closer to the end of the PROMOS R&D project, the company migrated the data center into a cloud service. This forced us to resolve new technological issues, related with a different access to data. Nevertheless, it also allowed us, for the first time, to work with the entire DSP data (and not just sampled data). These data access changes, produced through the three year execution of the PROMOS R&D project, resulted in distinct datasets that were analyzed through time. For instance, the data collection period used in (*Matos et al.*, 2018) is distinct from the one used in (*Matos et al.*, 2019a). On one hand, this can be viewed as a limitation, since it can be viewed as a lack of coherence of the research design, as each published work tended to approach different datasets. On the other hand, this can also be viewed as a strength, since the proposed methods, particularly the PCP transform reuse learning DFFN was constantly tested on different datasets through time. Given that it obtained systematically high quality results, this type of dynamic research design attests the robustness of the proposed method.

It should be noted that the analyzed DSP platform often changes, thus at each time period when the research was executed, it made sense to explore the latest features of the platform. For instance, the city feature was not initially available. We only had access to such feature when performing the work the DL binary user CVR prediction work (*Matos et al.*, 2019a). And later on, when we obtained the entire DSP data from the cloud, the city feature was not available anymore.

The DSP company was particularly interested in the results obtained in the second (*Matos et al.*, 2019a) and fourth works (*Matos et al.*). During the PhD work execution it was not possible to deploy the proposed IDSS in a real DSP platform. Nevertheless, the developed algorithms, including the IDSS R and Python code, was provided to OLAmobile, which managed to implement the algorithm in their DSP. It should be also mentioned that the proposed methods and models created throughout this PhD thesis have been adapted and implemented to other R&D projects. In particular, we explored the PCP transform and several DFFN models in: several use cases of the **Factory of the Future (FOF)** project, which involves the Bosch Car Multimedia company; and **TexBoost** project (`https://www.texboost.pt/`). Finally, it should be highlighted that the categorical transformations (including the PCP encoding) were implemented in the Cane python module (*Matos et al.*, 2020), which has already obtained $38,525$ downloads[1] since its release in June 3^{rd} of 2020.

1 According to the website: `https://pepy.tech/project/cane`.

4.2 Future Work

There are several interesting future research directions. For instance, since the DSP data is highly unbalanced, it may be interesting to approach the binary user CVR prediction task by using one-class ML models (e.g., Isolation Forests, AutoEncoders). These models can be trained only with normal ("No Sale" events) and then a sale could be predicted as an anomaly. Another interesting research direction could rely on the usage of Generative Adversarial Network (GAN) to augment the less represented classes, generating more synthetic examples with positive labels (e.g., "Sale"). Furthermore, the ordinal classification results were considered unsatisfactory by the DSP company. In future work, the ordinal results could be improved by adopting a multi-objective (F1-score for each class) evolutionary learning to train a MLP model. Moreover, eXplainable Artificial Intelligence (XAI) methods, such as provided by LIME (*Ribeiro et al.*, 2016) and SHAP (*Lundberg and Lee*, 2017), could be used to extract understandable knowledge for the DSP domain experts. Such knowledge could be potentially useful to detect which ads perform better for some types of users. In addition, the proposed PCP categorical transform provided an interesting value, resulting in simpler DL models that performed well while requiring a reduced computational effort (in terms of memory and processing time). Thus, there is a potential value in applying the same transform to other ML application domains that also contain high cardinality features.

REFERENCES

Amado, A., P. Cortez, P. Rita, and S. Moro, Research trends on Big Data in Marketing: A text mining and topic modeling based literature analysis, *European Research on Management and Business Economics*, *24*(1), 1–7, 2018.

Arnott, D., and G. Pervan, A critical analysis of decision support systems research revisited: The rise of design science, *Journal of Information Technology*, *29*(4), 269–293, doi: 10.1057/jit.2014.16, 2014.

AVINetworks, What is Infrastructure as a Service (IaaS)? Definition & FAQs, 2021.

Banker, K., *MongoDB in action*, Manning Publications Co., 2011.

Barone, A., and M. James, Digital Marketing Definition, 2020.

Barros, S., What is a DSP (Demand-Side Platform)? - Mobidea Academy, 2017.

Batista, G. E., R. C. Prati, and M. C. Monard, A study of the behavior of several methods for balancing machine learning training data, *ACM SIGKDD explorations newsletter*, *6*(1), 20–29, 2004.

Bengio, Y., Learning deep architectures for ai, *Found. Trends Mach. Learn.*, *2*(1), 1–127, doi: 10.1561/2200000006, 2009.

Bifet, A., G. Holmes, R. Kirkby, and B. Pfahringer, MOA: massive online analysis, *Journal of Machine Learning Research*, *11*, 1601–1604, 2010.

Bifet, A., R. Gavaldà, G. Holmes, and B. Pfahringer, *Machine learning for data streams: with practical examples in MOA*, MIT press, 2018.

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

Bureau, I. A., How an Ad is Served with Real Time Bidding (RTB) - IAB Digital Simplified - YouTube, 2014.

Campitelli, G., and F. Gobet, Herbert Simon's Decision-Making Approach: Investigation of Cognitive Processes in Experts, *Review of General Psychology*, *14*(4), 354–364, doi: 10.1037/a0021256, 2010.

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

Chapman, P., J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, Crisp-Dm 1.0, *CRISP-DM Consortium*, p. 76, 2000.

Chawla, N. V., K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, SMOTE: Synthetic Minority Over-sampling Technique, *Journal of Artificial Intelligence Research*, *16*, 321–357, 2002.

Chen, T., and C. Guestrin, Xgboost: A scalable tree boosting system, in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pp. 785–794, ACM, 2016.

Chen, Y., D. Pavlov, and J. F. Canny, Large-scale behavioral targeting, in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, p. 209, ACM Press, New York, New York, USA, 2009.

Chen, Y., Y. Chen, Y. Hsu, and J. Wu, Predicting consumers' decision-making styles by analyzing digital footprints on facebook, *International Journal of Information Technology and Decision Making*, *18*(2), 601–627, 2019.

Chollet, F., *Deep Learning with Python*, Manning Publications Co., USA, 2017.

Corporation, M., and S. Weston, *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*, r package version 1.0.16, 2020.

Cortez, P., Data Mining with Neural Networks and Support Vector Machines using the R/rminer Tool, in *Advances in Data Mining – Applications and Theoretical Aspects, 10th Industrial Conference on Data Mining*, edited by P. Perner, pp. 572–583, LNAI 6171, Springer, Berlin, Germany, 2010.

Cortez, P., *rminer: Data Mining Classification and Regression Methods*, 2016.

Cox, D. R., The regression analysis of binary sequences (with discussion), *J Roy Stat Soc B*, *20*, 215–242, 1958.

David, C., J. Perrey, T. McGuire, J. Gordon, and D. Spillecke, *Marketing & Sales Big Data , Analytics , and the Future of Marketing and Sales*, March, Mckinsey & Company, 2015.

Dean, J., and S. Ghemawat, Mapreduce: simplified data processing on large clusters, *Commun. ACM*, *51*(1), 107–113, 2008.

Deng, L., J. Gao, and C. Vuppalapati, Building a Big Data Analytics Service Framework for Mobile Advertising and Marketing, 2015.

Dhar, V., Data science and prediction, *Commun. ACM*, *56*(12), 64–73, 2013.

Du, M., R. State, M. Brorsson, and T. Avenesov, Behavior profiling for mobile advertising, *Proceedings of the 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies - BDCAT '16*, pp. 302–307, 2016.

Eckerson, W., Predictive Analytics | Transforming Data with Intelligence, 2007.

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

Frank, E., and M. A. Hall, A simple approach to ordinal classification, in *Machine Learning: EMCL 2001, 12th European Conference on Machine Learning, Freiburg, Germany, September 5-7, 2001, Proceedings*, pp. 145–156, 2001.

Frankenfield, J., Cloud Computing Definition, 2020.

Friedman, J. H., Greedy function approximation: A gradient boosting machine, *The Annals of Statistics*, *29*(5), 1189–1232, 2001.

Garg, N., *Apache Kafka*, 1–74 pp., 2013.

Gautier, L., rpy2 Documentation Release 2.8.4 Laurent Gautier rpy2 contributors, 2016.

Goodfellow, I., Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

Gubela, R. M., A. Bequé, S. Lessmann, and F. Gebert, Conversion uplift in e-commerce: A systematic benchmark of modeling strategies, *International Journal of Information Technology and Decision Making*, *18*(3), 747–791, 2019.

Guo, H., R. Tang, Y. Ye, Z. Li, X. He, and Z. Dong, Deepfm: An end-to-end wide & deep learning framework for ctr prediction, *arXiv preprint arXiv:1804.04950*, 2018.

Guo, H., B. Chen, R. Tang, W. Zhang, Z. Li, and X. He, An embedding learning framework for numerical features in ctr prediction, in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2021, August 14-18, 2021*, ACM, 2021.

Hinton, G., and R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, *science*, *313*(5786), 504–507, 2006.

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

Hulten, G., L. Spencer, and P. Domingos, Mining time-changing data streams, *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01*, *18*, 97–106, 2001.

IBM, What is distributed computing - IBM Documentation, 2014.

Igelnik, B., and J. M. Zurada, *Efficiency and Scalability Methods for Computational Intellect*, 1st ed., IGI Global, Hershey, PA, USA, 2013.

Janardan, and S. Mehta, Concept drift in streaming data classification: Algorithms, platforms and issues, *Procedia Computer Science*, *122*, 804–811, doi: https://doi.org/10.1016/j.procs.2017.11.440, 5th International Conference on Information Technology and Quantitative Management, ITQM 2017, 2017.

Jones, E., T. Oliphant, P. Peterson, et al., SciPy: Open source scientific tools for Python, 2001.

Jones, M. T., Deep learning architectures, 2017.

Jr., L. R. F., and S. M. Johnson, A tournament problem, *The American Mathematical Monthly*, *66*(5), 387–389, doi: 10.1080/00029890.1959.11989306, 1959.

Kaufmann, M., Chapter 10 - cuda in a cloud and cluster environments, in *CUDA Application Design and Development*, edited by R. Farber, pp. 241–264, Boston, 2011.

Kingma, D. P., and J. L. Ba, Adam: A method for stochastic optimization, in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, International Conference on Learning Representations, ICLR, 2015.

LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, Backpropagation applied to handwritten zip code recognition, *Neural computation*, *1*(4), 541–551, 1989.

LeCun, Y., Y. Bengio, and G. Hinton, Deep learning, *nature*, *521*(7553), 436, 2015.

Li, F.-F., et al., CS231n: Convolutional Neural Networks for Visual Recognition, 2021.

Lu, Q., S. Pan, L. Wang, J. Pan, F. Wan, and H. Yang, A practical framework of conversion rate prediction for online display advertising, in *Proceedings of the ADKDD'17, Halifax, NS, Canada, August 13 - 17, 2017*, pp. 9:1–9:9, 2017.

Lunardon, N., G. Menardi, and N. Torelli, ROSE : A Package for Binary Imbalanced Learning, *The R Journal*, *6*(June), 79–89, 2014.

Lundberg, S. M., and S.-I. Lee, A unified approach to interpreting model predictions, in *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, pp. 4765–4774, Curran Associates, Inc., 2017.

Matos, L. M., P. Cortez, R. Mendes, and A. Moreau, A deep learning based decision support system for mobile performance marketing, *Submitted to a journal.*

Matos, L. M., P. Cortez, R. Mendes, and A. Moreau, 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*, pp. 140–146, 2018.

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

Matos, L. M., P. Cortez, R. C. Mendes, and A. Moreau, 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*, *Lecture Notes in Computer Science*, vol. 11871, pp. 60–67, Springer, 2019b.

Matos, L. M., P. Cortez, and R. C. Mendes, Cane - categorical attribute transformation environment, 2020.

Meier, A., and M. Kaufmann, *SQL & NoSQL Databases: Models, Languages, Consistency Options and Architectures for Big Data Management*, 238 pp., 2019.

Merriman, D., E. Horowitz, and K. Ryan, MongoDB, 2007.

Mikalef, P., I. O. Pappas, J. Krogstie, and M. Giannakos, Big data analytics capabilities: a systematic literature review and research agenda, doi: 10.1007/s10257-017-0362-y, 2017.

Monnappa, A., The History and Evolution of Digital Marketing [Updated], 2021.

Moolayil, J., *Learn Keras for Deep Neural Networks*, Apress, doi: 10.1007/978-1-4842-4240-7, 2019.

Moro, S., P. Cortez, and P. Rita, A data-driven approach to predict the success of bank telemarketing, *Decision Support Systems*, *62*, 22 – 31, doi: https://doi.org/10.1016/j.dss.2014.03.001, 2014.

Moro, S., P. Cortez, and P. Rita, 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, doi: 10.1007/s00521-014-1703-0, 2015a.

Moro, S., P. Cortez, and P. Rita, Feature Selection Strategies for Improving Data-Driven Decision Support in Bank Telemarketing, Ph.D. thesis, ISCTE, 2015b.

Nyce, C., Predictive Analytics White Paper, 2013.

Okewu, E., P. Adewole, and O. Sennaike, Experimental comparison of stochastic optimizers in deep learning, in *Computational Science and Its Applications - ICCSA 2019 - 19th International Conference, Saint Petersburg, Russia, July 1-4, 2019, Proceedings, Part V*, *Lecture Notes in Computer Science*, vol. 11623, pp. 704–715, Springer, 2019.

OLAmobile, O., Olamobile, 2011.

Oliphant, T., NumPy: A guide to NumPy, USA: Trelgol Publishing, 2006.

Oliveira, N., P. Cortez, and N. Areal, 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, 2017.

Oza, N. C., Online bagging and boosting, in *2005 IEEE international conference on systems, man and cybernetics*, vol. 3, pp. 2340–2345, Ieee, 2005.

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

Power, D. J., Using 'Big Data' for analytics and decision support, *Journal of Decision Systems*, *23*(2), 222–228, 2014.

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

Ramírez-Gallego, S., B. Krawczyk, S. García, M. Wozniak, and F. Herrera, A survey on data preprocessing for data stream mining: Current status and future directions, *Neurocomputing*, *239*, 39–57, 2017.

Ribeiro, M. T., S. Singh, and C. Guestrin, "why should I trust you?": Explaining the predictions of any classifier, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pp. 1135–1144, 2016.

Rossum, G., Python reference manual, *Tech. rep.*, Amsterdam, The Netherlands, The Netherlands, 1995.

Rouse, M., What is decision support system (DSS)? - Definition from WhatIs.com, 2010.

Shankar, V., and S. Balasubramanian, Mobile marketing: A synthesis and prognosis, *Journal of Interactive Marketing*, *23*(2), 118 – 129, anniversary Issue, 2009.

Shannon, C. E., A mathematical theory of communication, *The Bell system technical journal*, *27*(3), 379–423, 1948.

Shmueli, G., and O. R. Koppius, Predictive analytics in information systems research, *MIS Quarterly*, *35*(3), 553–572, 2011.

Simon, H. A., The new science of management decision., 1960.

Sousa, R. G., I. Yevseyeva, J. F. P. da Costa, and J. S. Cardoso, Multicriteria models for learning ordinal data: A literature review, in *Artificial Intelligence, Evolutionary Computing and Metaheuristics - In the Footsteps of Alan Turing*, pp. 109–138, 2013.

Sprague Jr, R. H., A framework for the development of decision support systems, *MIS quarterly*, pp. 1–26, 1980.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, *The Journal of Machine Learning Research*, *15*(1), 1929–1958, 2014.

Sterling, T., M. Anderson, and M. Brodowicz, Chapter 19 - mapreduce, in *High Performance Computing*, edited by T. Sterling, M. Anderson, and M. Brodowicz, pp. 579–589, Morgan Kaufmann, Boston, 2018.

Sutskever, I., J. Martens, G. Dahl, and G. Hinton, On the importance of initialization and momentum in deep learning, in *30th International Conference on Machine Learning, ICML 2013*, PART 3, pp. 2176–2184, 2013.

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

Teixeira, S., J. Martins, F. Branco, R. Gonçalves, M. Au-Yong-Oliveira, and F. Moreira, A Theoretical Analysis of Digital Marketing Adoption by Startups, pp. 94–105, Springer, Cham, 2018.

Torgo, L., *Data Mining with R, learning with case studies*, Chapman and Hall/CRC, 2010.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, Attention is all you need, *arXiv preprint arXiv:1706.03762*, 2017.

Volkova, S., Data Stream Mining : A Review of Learning Methods and Frameworks Learning from Streaming Text Data, pp. 1–7, 2012.

Wen, H., J. Zhang, Y. Wang, F. Lv, W. Bao, Q. Lin, and K. Yang, Entire space multi-task modeling via post-click behavior decomposition for conversion rate prediction, in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, edited by J. Huang, Y. Chang, X. Cheng, J. Kamps, V. Murdock, J. Wen, and Y. Liu, pp. 2377–2386, ACM, 2020.

Witten, I. H., E. Frank, M. A. Hall, and C. J. Pal (Eds.), *Data Mining (Fourth Edition) Practical Machine Learning Tools and Techniques*, fourth edition ed., Morgan Kaufmann, 2017.

Wu, K.-W., et al., A Two-Stage Ensemble of Diverse Models for Advertisement Ranking in KDD Cup 2012, 2012.

Yang, H., and Q. Lu, Large Scale CVR Prediction through Dynamic Transfer Learning of Global and Local Features, in *BIGMINE*, vol. XX, pp. 1–16, 2016.

Yoshikawa, Y., and Y. Imai, A Nonparametric Delayed Feedback Model for Conversion Rate Prediction, 2018.

Zeiler, M. D., ADADELTA: an adaptive learning rate method, *CoRR*, *abs/1212.5701*, 2012.

Zhang, H., The Optimality of Naive Bayes Naive Bayes and Augmented Naive Bayes, *American Association for Artificial Intelligence*, 2004.

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

Zhang, W., T. Du, and J. Wang, Deep learning over multi-field categorical data - A case study on user response prediction, in *Advances in Information Retrieval - 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20-23, 2016. Proceedings*, pp. 45–57, 2016.

PROMOS – Previsão e otimização de campanhas publicitárias Para Dispositivos móveis em modelo de Subscrição, NORTE-01-0247-FEDER-017497 All rights reserved.