Marketing Campaigns Optimisation with Artificial Neural Networks and Genetic Algorithms

By
Saad Chaouki

Dissertation Presented for the Degree of

MSc in Business Analytics

2017/18

Abstract

The last few years have witnessed a significant increase in the revenues generated by the online advertisement industry. In fact, it has become a crucial feature of economic life and drove large corporations to focus solely on it. Before launching a marketing campaign, advertisers aim at finding the appropriate parameters to reach the right audience and maximise their returns for a given budget. In real-time bidding, where the ads slots are sold and purchased in real-time, advertisers have to bid proportionally to the predicted outcome. In that sense, they need to wait for the adequate users to bid and potentially show their ads which might lead to a delay in results if the right users do not visit the publishers' websites. This paper proposes a dynamic display advertisements creation methodology using a genetic algorithm and an artificial neural network to tackle this issue. Based on the user, the methodology finds the right ad parameter to set that maximise the probability of occurrence of a click. Thus, allowing the advertisers to adapt their ads to each user. The methodology is tested on a real-life dataset and shows a significant improvement in the predicted CTR for the newly constructed ads from 31% to 91% while being able to handle parameters constraints, new, and returning users. In-depth experimenting further shows that the methodology's advantages can be extended to select the appropriate audience and select the relevant parameters to a given audience. Making the methodology a valuable tool for the entire marketing campaigns decision making.

Keywords— Marketing Campaigns Optimisation, Artificial Neural Networks, Genetic Algorithms

Acknowledgements

I would like to thank Dr. De Smedt Johannes for his help and valuable suggestions throughout this dissertation.

My gratitude goes to my family and friends for their continuous support and encouragement.

Contents

List of Figures

List of Algorithms

List of Tables

List of Abbreviations

ANN Artificial Neural Network

AUC Area Under Curve

AUROC Area Under Receiver Operating Characteristic

BN Bayesian Network

CART Classification And Regression Trees

CR Conversion Rate

CTR Click-Through Rate

DNN Deep Neural Network

DT Decision Tree

FM Factorization Machines

GA Genetic Algorithm

K-NN K-Nearest Neighbors

KPI Key Performance Indicator

LR Logistic Regression

LSTM Long-Short Term Memory

ML Machine Learning

MLP Multi-Layer Perceptron

NB Naïve Bayes

RF Random Forest

RNN Recurrent Neural Networks

ROC Receiver Operating Characteristic

SGD Stochastic Gradient Descent

SMOTE Synthetic Minority Over-Sampling Technique

SVM Support Vector Machines

SVR Support Vector Regression

Chapter 1

Introduction

Online Advertisement is a crucial feature of economic life. Throughout the past few years, its influence on the advertising industry has been significant and enabled it to hold a large portion of the total advertising market (Goldfarb 2014). This has been stimulated by an increase in the time spent online driving companies to improve their online presence and even to rely solely on online advertisement (Goldfarb 2014). In fact, the total revenues generated by the industry in the US only has been consistently increasing from $22.7 billion in 2009 to $88 billion in 2017 (IAB 2017) eventually surpassing broadcast and cable television's revenues (Korula et al. 2016).

Similarly to traditional advertisements, the goal of online advertisements is convincing customers to purchase a product, familiarising customers about the features of a product, or complementing a product to increase its consumption (Goldfarb 2014). However, when referring to online advertisement, three categories are usually referred to. These categories are search, classified and display advertisement. Search advertisement is the advertisements, usually in form of text, that appears in search engines like Google or Bing. Classified advertisement is the advertisements on websites that are specialised to only display ads like Craigslist[1]. Finally, display advertisement includes text ads, videos, and banners that are shown on different websites such as Facebook[2] and constitutes their main source of revenue (Korula et al. 2016). Moreover, it is a major category of online advertisement that contributes to no less than 40% of the total online advertisement revenues and receives a considerable attention from researchers and practitioners who develop methods to optimise the way these ads are displayed (Korula et al. 2016). More specifically, they apply a method called micro-targeting where the ads can be displayed to users based on their characteristics (Miralles-Pechuán et al. 2018). In that sense, the focus of the methodology is on display advertisement.

1.1 Online Advertisement Eco-System

The usual online advertisement eco-system is composed of four main players. Namely, the advertisers, advertising networks, publishers, and users. The advertisers create a marketing campaign. Advertising networks serve as intermediaries between the advertisers and the publisher;they communicate the ads, restriction, and maximum budget to publishers. The publishers are the link between the users and the ads; they supply the

[1] American classified advertisement website
[2] American Social Media website

ads to the users that satisfy the requirements put by the advertisers (Evans 2008).

In real-time bidding where the advertising spots are sold on a real-time basis, this configuration is subject to some changes. In this case, different entities join the game which are Supply-Side Platforms (SSPs), Demand-Side Platforms (DSPs) and Ad Exchanges (Lin et al. 2016). The process goes as follow. (i) The advertiser provides the DSP with the ads to show and the bidding strategy. (ii) A user visits the publisher's website. (iii) If the publisher has a free spot for an ad, the information is sent to the DSP through the ad exchange network. (iv) The DSP computes the bid value based on the information retrieved from the publisher in step (iii) and the advertiser in step (i). (v) The bid is submitted and, based on an auction, the winner is determined and his ad is shown to the user (Lin et al. 2016). This is a fast process that happens in the matter of milliseconds (Zhang, Yuan & Wang 2014).

Because the advertisers are subject to budget constraints, the bids have to be relative to computed key performance indicators (KPIs) such as the Click-Through Rate (CTR) or the Conversion Rate (CR) (Zhang, Yuan & Wang 2014). Consequently, they have to wait for a specific kind of users to access the publishers' websites before bidding. In this scenario, one question arise:

- Is it possible for advertisers to dynamically build the right advertisement for each user visiting the website?

In this case, the challenge is for the advertiser to find the right parameters to adapt to the user's preferences.

1.2 Proposed Methodology

The proposed methodology is a dynamic advertisement building procedure that finds the right ad for each user. More precisely, the aim is to learn from previous click behaviour to be able to map a user-ad matching to a probability of click. The predicted values are then used in a Genetic Algorithm (GA) to find the most suitable ad to show to a user. Thus, this work sought to achieve the following:

- Train a machine learning (ML) model to accurately predict the CTR of a user-ad matching.
- Incorporate the predictor in a genetic algorithm (GA) to dynamically build advertisements and achieve higher CTRs.

The remainder of this paper is organised as follow. Chapter 2 reviews the various literature on the optimisation of marketing campaigns from different perspectives and the prediction of a crucial metric in online advertisements which is the CTR. Chapter 3 describes in detail the proposed methodology. Chapter 4 showcases the hyperparameters tuning process of the methodology as well as three different optimisation experiments. Chapter 5 describes the implications of the methodology from a research and business point of views. Lastly, Chapter 6 concludes the paper and suggests future work.

Chapter 2

Literature Review

This paper aims to optimise marketing campaign parameters on a user basis. More precisely, the goal is to develop a methodology that predicts the outcome of a given user-ad matching and improves it. Therefore, the focus of the literature review is on marketing campaigns optimisation from different facets as well as the prediction of an crucial metric in online advertising, namely, the CTR.

The CTR is the probability that a user clicks on a displayed advertisement (Graepel et al. 2010) with 1 being a click and 0 being a non-click. It is the primary metric employed to assess the quality of a user-ad matching. By predicting the CTR, it becomes possible to determine the most suitable advertisement to display to a user based on his characteristics; which is essential in advertising relevant products an services online (Yin et al. 2014) and plays a significant role in the online advertising industry (McMahan et al. 2013). To predict the CTR, machine learning (ML) has become the most influential field (Miralles-Pechuán et al. 2018). In fact, its evolution in the last years has had a significant positive impact on online advertising companies (Gabrilovich et al. 2009).

In that sense, Section 2.1 of the literature review covers the different facets of marketing campaigns optimisation while Section 2.2 focuses on the CTR and comprises the various methodologies authors suggest to predict it.

2.1 Marketing Campaigns Optimisation

Throughout the years, advertisers have tried to optimise the way they exhibit their marketing campaigns from different facets. Section 2.1.1 covers the direct optimisation of the CTR. While some authors do not directly optimise the CTR, they make use of it in their optimisation. Thus, bidding price optimisation in a real-time bidding context is discussed in Section 2.1.2. Section 2.1.3 discusses the optimisation of the reach of the campaigns by finding the right delivery channels. Naturally, some researchers focused on optimising the revenues and profits generated as well as managing the budget and will be discussed in Section 2.1.4. Finally, budget allocation is discussed in Section 2.1.5.

2.1.1 Click-Through Rate Optimisation

Various authors choose to directly optimise the CTR as a way to reach optimal marketing campaign configurations. To do so, the CTR is either predicted using one of the methodologies cited in Section 2.2 or assumed to be given. In both cases, the authors agree that users' data is crucial in this case. Langheinrich et al. showed in 1999 that adapting online advertisements to a user's interest instead of randomly has a significant impact on the average CTR. Thus, the authors developed a model that displays the best ads to maximise this one while adding a minimum number of displays constraint to each ad (Langheinrich et al. 1999). The use of a linear model for this problem was also suggested by Chickering & Heckerman a few years later in 2003. Chickering & Heckerman identify optimal delivery schedules based on attributes that aim to maximise the CTR. As for Langheinrich et al., they showed that this procedure of creating calendars for ad delivery leads to better results than a manually or randomly created one (Chickering & Heckerman 2003).

The methodology proposed by Langheinrich et al. was later improved by Nakamura & Abe in 2005 by looking into challenges related to CTR estimation and suggesting guidelines for a successful deployment of the model (Nakamura & Abe 2005). To further establish the importance of users' data, Yan et al. found that correctly targeting users in sponsored search leads to an increase of 670% in CTR. The reason is that users who clicked on one advertisement will have the same online behaviour (Yan et al. 2009); which can be leveraged to increase the accuracy of predictions. These findings were confirmed by several authors who deal with CTR maximisation. Recently, and because of a massive amount of data created in the last few years, the focus is on the ability to deal with large datasets and social media advertisement. In that sense, Chen et al. used MapReduce[1] to deal with the large-scale data used from Yahoo! website. The goal of the authors is to target the right users by maximising their CTR while making use of their characteristics and the marketing campaign's metadata (Chen et al. 2009). Abbassi et al. turns to the optimisation of display advertisement in social media websites. The authors argue that the probability of a click occurring is related to the number of 'friends' that have clicked on it. Thus, the authors develop a heuristic that creates a list of optimal ranked users to show the advertisement to (Abbassi et al. 2015). Likewise, Pepelyshev et al. address methods to maximise the CTR and determine whether to display an ad to a user or no based on his characteristics. The authors suggest an adaptive strategy that was tested and leads to an increase of 2.5 times the original CTR (Pepelyshev et al. 2015).

2.1.2 Bidding Price Optimisation

The importance of paying the right price for the ad to be delivered in a real-time bidding context has led to an increasing interest in bidding price optimisation. Perlich et al. combine the use of ML and game theory to determine the right amount to bid in a real-time bidding context (Perlich et al. 2012). Zhang, Yuan & Wang followed the same path by providing a bidding strategy that maximises the expected revenues subject to a budget constraint. The framework was compared to various bidding strategies and has proven to lead to better performance measures such as win and conversion rates (Zhang, Yuan & Wang 2014). Lin et al. in 2016 took into consideration the CTR. The authors included the CTR to make sure that it is taken into consideration in the bid price optimisation (Lin et al. 2016). Thus, the authors optimise both the bid price and the CTR together (Lin et al. 2016). The last few years have witnessed a major shift to reinforcement learning for the bid optimisation problem. In 2017, Cai et al. suggested the use of reinforcement learning

[1] programming paradigm for scalability across thousands of servers.

with neural networks to learn the optimal bidding strategy. The model was then tested using A/B testing and shows a superior performance (Cai et al. 2017). The number of studies in this direction have continued in 2018 where Wu et al. (Wu et al. 2018) and Jin et al. (Jin et al. 2018) also used reinforcement learning for bidding optimisation. Wu et al. introduce budget constraints in the model (Wu et al. 2018) while Jin et al. suggest the use of clustering to deal with large datasets while having a bidding agent for each cluster (Jin et al. 2018).

2.1.3 Reach Optimisation

Regarding reach optimisation, Danaher et al. published the first paper of its kind to optimise the campaigns by selecting the most appropriate social media websites as channels of distribution. The exposure of the ads was formalised as a mathematical program and then optimised to maximise the reach of each campaign by finding the right proportion of the budget to be allocated to each channel (Danaher et al. 2010). However, Danaher et al.'s methodology is unable to deal with a large choice of websites. In 2015, Paulson et al. tackled this limitation and increased the scale of selection to thousands of websites while including other features such as demographics (Paulson et al. 2015). Bharadwaj et al. looks at the reach optimisation from a time perspective. They propose an algorithm that creates efficient allocation plans for advertisements while taking into consideration the time factor. The authors optimise the reach by finding the right fraction of ads to be displayed at a specific time (Bharadwaj et al. 2012).

2.1.4 Revenues and Profits Optimisation

The revenues and profit optimisation get a considerable share of the literature on advertisements optimisation. In 2012, Aksakallı maximised the revenues of advertisers in a display advertisement context. The author's novelty is that he explicitly incorporated the location of the ad and its content into an integer program. Results of his experimenting show an increase in the revenues of advertisers using the same campaign budget (Aksakallı 2012). Mookerjee et al. share the same goal with Aksakallı but add CTR constraint to the model. The authors use a Logistic Regression (LR) as the method of choice to predict the CTR. The CTR constraint ensures that only the most relevant ads are shown to users without filling the channels with irrelevant ads (Mookerjee et al. 2012). Predicting the CTR before optimising the campaign was also done by Ren et al. who used LR as well. However, the difference between Ren et al. and Mookerjee et al. is that Ren et al. use the predicted CTR to compute a bid value as to maximise the revenues (Ren et al. 2016). While optimising the campaigns, the parameters for the CTR predictor are learned and adjusted. A/B testing has revealed an increase in profits of 25.5% (Ren et al. 2016). Zhang et al. and Ren et al. turn to bid price as a way to maximise the profit (Ren et al. 2018). Zhang et al. jointly optimise the bid price and the budget of the campaign to maximise the advertiser's revenue; they aim to maximise the profit by optimising the price to bid. The joint optimisation gives better results and increases the revenue of advertisers (Zhang et al. 2012). Ren et al. propose a different methodology from his previous one in 2016 by taking into consideration the utility (CTR) of the ad as well as its cost. These two are then used to find the optimal bidding price that will maximise the profit of the campaign. (Ren et al. 2018).

2.1.5 Budget Distribution Optimisation

Budget distribution aims at finding the optimal way to distribute a given budget for a consistent impact. To do so, Muthukrishnan et al. formulate a stochastic version of their methodology by taking into consideration

the distribution of future queries (Muthukrishnan et al. 2007). Amin et al., on the other hand, focus on maximising the number of clicks generated by a specific budget by casting the optimisation problem as a Markov Decision Process. Their methodology was tested on real-life auction data and compared to several other models including Q-learning (reinforcement learning). The methodology outperforms the various models and converges faster (Amin et al. 2012). Lee et al. focus on optimising the CTR and conversion rate while making sure that the budget is consistently distributed over time. The methodology's simplicity allows it to handle millions of requests per second while improving the CTR and CR. The CTR is, however, assumed to be given (Lee et al. 2013). Agarwal et al. suggest a different way of budget pacing. The authors use an algorithm that forecasts the traffic at different times of the day and adjusts the budget accordingly. The algorithm gave positive results before being fully deployed at LinkedIn[2] (Agarwal et al. 2014). Xu et al. aim at solving the same problem but suggest a smart pacing heuristic that learns from historical data to control the spacing of ads online. The methodology achieves smooth pacing and boosts campaign performance measured by a reduction in the generated Cost per Click (CPC) (Xu et al. 2015). In 2012, and because of the NP-hard nature of the optimisation, Deane proposed a hybrid methodology to solve the budget scheduling problem in online advertisement. The author uses a metaheuristic (Genetic Algorithms) with Augmented Neural Networks that proved to be effective (Deane 2012).

2.1.6 Summary

As specified previously, ML has had a significant positive impact on online advertising companies during the past few years (Gabrilovich et al. 2009). To visualise the impact on the research, Figure 2.1 showcases the number of papers published per year, goal, as well as the use of ML either as the principal methodology or to predict the CTR. The list of papers is summarised in Appendix A.

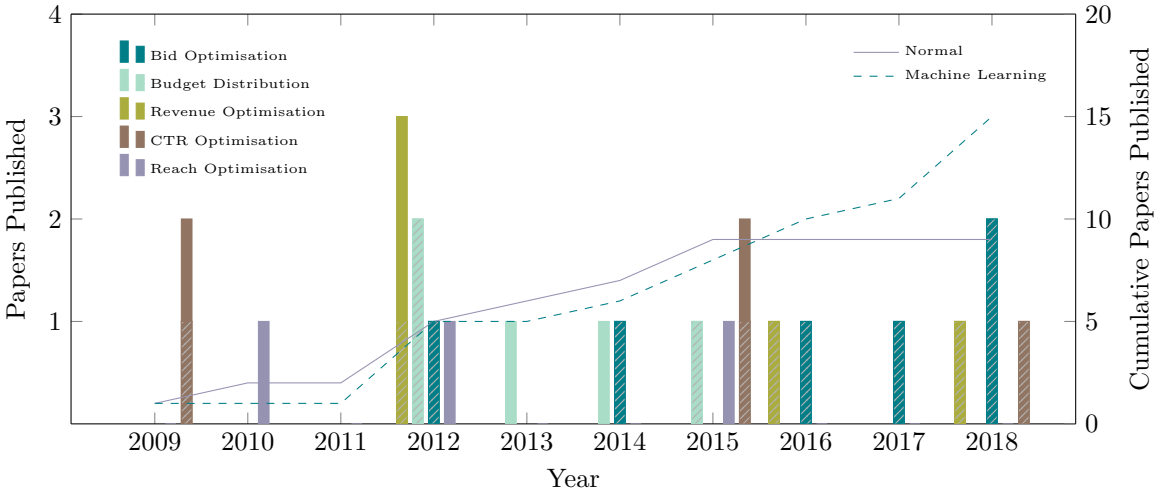

Figure 2.1: Frequency of Papers' Publication for Marketing Campaign Optimisation. Visual representation of the papers cited in Section 2.1 starting 2009 classified by goal. The dashed parts of the graph represent the proportion of papers that make use of ML methodologies.

While the number of papers published per year about the various facets of the optimisation is consistent, the research community is increasingly using ML; its improvement makes it possible to investigate new aspects of the research. However, the increase in the use of ML is driven by the bid optimisation research

[2]business and employment-oriented service.

while other areas seem to be lacking in that sense. More importantly, even though the CTR can be used in the various optimisation facets, only one paper is found that optimises it using ML since 2015. Thus, the research community seems to focus more on the prediction of the CTR without developing methodologies to maximise it.

Moreover, the majority of researchers use mathematical programming to solve their optimisation problems. Consequently, there seems to be a gap in the use of metaheuristics for CTR optimisation similar to Deane's methodology for the budget distribution problem. Miralles-Pechuán et al. in 2018 tackle this gap by suggesting a combination of a GA and LR to predict the CTR and find an adequate marketing campaign configuration. The LR is used to predict the CTR while the GA optimises the marketing campaign configuration. The methodology was tested on a dataset of mobile display advertisement campaign and proved to have a quick implementation and be cost-effective. Additionally, because the GA makes use of a fitness function, it can be easily adapted to optimise multiple facets such as bid optimisation (Miralles-Pechuán et al. 2018).

Consequently, CTR optimisation is the principal interest of this paper. Because of its novelty and ability to adapt to diverse uses, the goal is to build on Miralles-Pechuán et al.'s methodology from two sides. The first side is the use of a more appropriate model for CTR prediction as discussed in Section 2.2. The second side is to make the optimisation a user-based one where each user is subject to an optimisation procedure to find the most suitable ad. Thus, making sure that each user is entirely satisfied with the set of ads he is shown while using a methodology that can easily be adapted for different optimisation targets.

2.2 Click-Through Rate Prediction

As pointed out in Section 2.1, the most important methodology in CTR prediction is ML. The number of models in that scope is massive. However, in a CTR context, only a few models are mostly used. These models include but are not limited to Linear and Logistic Regression in Section 2.2.1, Classification and Regression Trees (CART) and Bayesian Networks (BN) in Section 2.2.2, Factorization Machines (FM) in Section 2.2.3, as well as Artificial Neural Networks in Section (ANN) 2.2.4. The difference with the methodologies discussed in Section 2.1 is the goal of the researchers. In Section 2.1 the goal of the research is to optimise the campaigns while the research presented in this section covers its prediction.

2.2.1 Linear and Logistic Regressions

LR is one of the most encountered and known ML methods in the CTR prediction literature (Chen et al. 2016). Kumar et al. used a simple LR to predict the CTR of ads by only using the impression of the ad (number of times it was displayed) and its position. The model achieved a 90% accuracy on a 25 GB dataset (Kumar et al. 2015). Kondakindi et al., however, included more predictors in the LR model to predict the CTR and compared its accuracy to NB and Vowpal Wabbit[3]. LR with stochastic gradient descent (SGD) and weight regularisation led to the best results (Kondakindi et al. 2014). In 2018, Dhanani & Rana confirmed the positive impact of stochastic gradient ascent (SGA) for parameters determination with LR by predicting the CTR based on characteristics such as the age, gender, keywords of the advertisements and query. The authors compared the performance of the model with SGA to a model with batch gradient ascent (BGA). The SGA model achieved better accuracy and training time (Dhanani & Rana 2018). The effectiveness of LR on a massive dataset was also confirmed in 2015 by Chapelle et al.. The authors presented a highly scalable, simple to use and to deploy, method based on LR. The model was tested on a terabyte of data and proved to be accurate (Chapelle et al. 2015). Some authors explored CTR prediction for new ads in particular. Richardson et al. proved that it is possible to predict the CTR of new ads by including in the LR model features about the ad itself, related terms, and source advertiser. Leading to a 30% decrease in error rate compared to the baseline model (Richardson et al. 2007).

Some other types of regressions were also studied. Agarwal et al. proposed a dynamic linear regression model. In the same way in which Kumar et al. added locations to the LR (Kumar et al. 2015), Agarwal et al. added the locations as well as the time factor to model the fatigue of users. Thus, improving the total accuracy of the model (Agarwal et al. 2009). More recently and in the same context, Effendi & Ali added keywords to include the context of advertisers using linear regression. However, this backfired by decreasing the accuracy from 95% to 83% but helps to compute the CTR from a different perspective (Effendi & Ali 2016). Similarly, Yin et al. exploited contextual features using a multi-task linear regression (Yin et al. 2014). The authors highlighted the difference between micro and macro factors affecting the CTR. On the micro level, some of the features included are the depth of the ad and the query. On the macro level, the authors studied the correlation of clicks in sponsored and organic searches (Yin et al. 2014). The authors concluded that including the micro and macro factors leads to an increase in the effectiveness of their model (Yin et al. 2014). A multivariate linear regression (Gao & Gao 2013) and multi-criteria linear regression (Wang et al. 2013) are also suggested. Gao & Gao focus on finding features that affect the CTR of some ads and use them to predict the CTR of similar ads. The features proved to be accurate, but some of them

[3]fast out-of-core learning system.

remained not understandable (Gao & Gao 2013). Wang et al. tested the multi-criteria linear model on a dataset provided by a Chinese internet company. It was then compared to Support Vector Regression (SVR) and LR and proved to be as efficient as these models based on the ROC and the AUC (Wang et al. 2013).

2.2.2 Classification and Regression Trees and Bayesian Networks

Classification and Regression Trees (CART) and Bayesian Networks (BNs) are also frequently used to predict the CTR. König et al. proposed a multi-additive regression tree to estimate the CTR, particularly for news-related queries. The model was found to be highly accurate (König et al. 2009). Trofimov et al. also suggested the use of boosted trees. The method achieved a better accuracy than linear and logistic regressions as well as gradient boosting machine (Trofimov et al. 2012). The combination of regression trees and LR was also explored as a CTR predictor by He et al. in 2014. The authors found that the combination of boosted CART and LR is a powerful one. The model was benchmarked against each method individually and outperformed them (He et al. 2014).

Regarding BN, Graepel et al. made use of the model to illustrate the relationship between different features and the probability of occurrence of a click in the Bing search engine. The model proved to be more accuracy than NB (Graepel et al. 2010). Fang et al. also makes use of a BN with a focus on new ads. The authors build a keyword BN and then use related keywords in new ads to predict their CTR (Fang et al. 2014) which is similar to Gao & Gao's methodology using linear regression (Gao & Gao 2013). The model gives encouraging results on a small dataset but requires more practical experimentation (Fang et al. 2014).

The results from a set of research show the accuracy of the regression models in CTR prediction. However, these models rely on the design of the features without taking into consideration the interactions between them. In other words, these models are unable to make use of the complete meaning provided by the dataset (Bengio et al. 2009, Rendle 2010, Zhang, Dai, Xu, Feng, Wang, Bian, Wang & Liu 2014). Thus, regression models are deficient when it comes to a complete and accurate CTR prediction (Rendle 2010, Jiang 2016). To tackle this issue, some authors propose different models that predict the CTR accurately while taking into consideration the interactions. Namely, Factorization Machines (FM) and Deep Neural Networks (DNN). FMs are models that combine Support Vector Machines (SVMs) and Factorization models. Thus, FMs consider the interactions between the different features (Rendle 2010). DNN is a type of ANNs (ANN) that has multiple hidden layers. The layers give it the possibility to represent different levels of abstraction and interaction in the data (LeCun et al. 2015). Thus, both FMs and DNNs are good candidates for CTR prediction.

2.2.3 Factorization Machines

McMahan et al. showed in 2013 several improvements generated by a Follow-The-Regularized-Leader-Proximal (FTRL-Proximal) algorithm in the context of a CTR prediction system. The algorithm shows excellent convergence and sparsity properties (McMahan et al. 2013). These findings were put to practice by Ta in 2015 by combining the FTRL-Proximal and FMs. Indeed, the combination outperformed the initial method which is FMs with SGD (Ta 2015). Some authors proposed variants of FMs. In 2016, Juan et al. suggested a Field-aware Factorization Machines (FFMs) for CTR predictions. FFMs differentiate between interactions occurring between features in the same field and features in different fields. The model out-

performs FMs (Juan et al. 2016) but still has a high number of parameters (Pan et al. 2018). Because of that, Pan et al. proposes in 2018 a Field-weighted Factorization Machines that models the interactions more efficiently than FFMs for CTR prediction. The model was tested and proved to be accurate and competitive to FFMs with only 4% of the parameters while slightly improving the performance measured by the AUC (Pan et al. 2018).

However, even though FMs and their variants show promising CTR results, it is highly complex. Because of that, only order two feature interactions are taken into consideration in practice (Guo et al. 2017); which might not always be sufficient to capture the information provided in a dataset.

2.2.4 Artificial Neural Networks

DNNs for CTR prediction are among the latest additions to the body of literature. Miralles-Pechuán et al. used DNN for CTR prediction. The model was compared to various ML models such as boosted classification trees, SVM, LR, and stochastic gradient boosting. The proposed DNN achieved better accuracy than the other models (Miralles-Pechuán et al. 2017). Chen et al. explored the use of DNNs to predict the CTR of image displayed ads. The author uses the pixels of the image and various other features. Testing over a dataset of 50 million records showed that the methodology is both accurate and efficient (Chen et al. 2016). In the same way, Covington et al. use DNNs to predict the CTR of videos on Youtube[4]. The predicted values were then ranked to find the videos to show to users. The DNN was tested using A/B testing and shows an increase in the watch time which is explained by an increase in the accuracy of Youtube's recommender system (Covington et al. 2016). Cheng et al. highlighted the difference between wide and deep learning models. A classical deep-only model is used to take into consideration the interactions between the features. A wide-only model, on the other hand, is used to memorise sparse features. The authors show that a wide-deep learning model outperforms the wide-only and deep-only models (Cheng et al. 2016). Qu et al. developed a product-based neural network. The model includes characteristics of products as a layer in the DNN architecture. This addition increases the accuracy of CTR predictions (Qu et al. 2016). Similarly, Zhou et al. develop in 2017 a model that captures the users' interest over time regarding certain ads. The model proved to be accurate and was deployed as the main system of display advertisement in Alibaba[5] (Zhou et al. 2017). Very recently, Jiang et al. improved the accuracy of regular DNN for CTR by combining it with a fuzzy theory (Jiang et al. 2018). The use of recurrent neural networks (RNNs) has also proved to be effective by Chen et al. in 2018. Their model was compared with LR, RF, and NB and provides better results based on the AUC (Chen et al. 2018).

While some authors explored the power of DNN individually, other authors tried to improve its performance by including advantages that other models offer. Jiang combined DNN with LR for CTR predictions. The DNN architecture is used to extract complex and abstract characteristics from the dataset that the LR is not able to obtain. This last one is then used to compute the CTR of a particular ad for a specific user. The author compared the model to a precise deep learning architecture (SAELR) and outperformed this one (Jiang 2016). However, the model still needs to be tested against more popular models for CTR prediction. Guo et al. combined DNN with FMs and used the Wide and Deep DNN concept proposed by Cheng et al.. The factorisation machines are used as a wide module to explore different features while the DNN is used

[4]American video-sharing website
[5]Chinese multinational e-commerce, retail, Internet, AI and technology conglomerate

to extract the interactions. The DeepFM model was found to be similar to most deep models in accuracy for CTR predictions (Guo et al. 2017).

2.2.5 Summary

Among the different methodologies, DNNs remain a robust method to learn the complex interactions between the features (Guo et al. 2017, Chen et al. 2016). Research shows that DNNs have various advantages over other more classical ML algorithms (Bengio et al. 2006). This strength resides in its backpropagation algorithm that adjusts the parameters to offer minimum differences between the predictions and the real data (Miralles-Pechuán et al. 2017). In practice, DNN was used in several competitions to predict the CTR and proved to be more or as efficient as LR, CART, FMs, and different other models (Chen et al. 2016). In addition to its capacity to capture interactions, the methodology proposed by Cheng et al. of Wide&Deep allows it to increase its scope and also capture sparse features (Chen et al. 2016). Because of these advantages, the research community has turned its focus from more classical methods of CTR prediction to ANNs. Appendix B summarises the different papers cited for CTR prediction, the publication year, and the methodology used. It is possible to detect an increase in the research around ANN for CTR prediction starting 2015. Simultaneously, the number of papers published in the last years using LR is decreasing. To better visualise the transition in the research to ANN methodologies, Figure 2.2 showcases an assessment of the literature around CTR prediction.

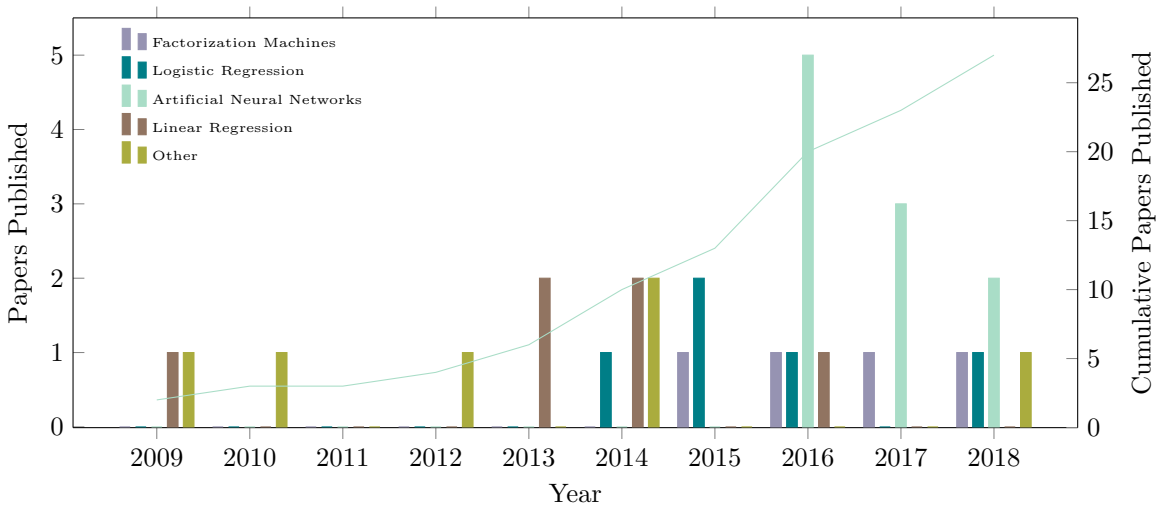

Figure 2.2: Frequency of Papers' Publication for CTR prediction by Methodology. Visual representation of the papers cited in Section 2.2 starting 2009. Artificial Neural Networks include normal Artificial Neural Networks as well as Deep Neural Networks and Recurrent Neural Network. Other includes the use of CART and Bayesian Networks for CTR prediction. Hybrid methods are represented twice in the methodologies they make use of.

Figure 2.2 showcases two essential things related to CTR prediction. The first one is the gradual increase in interest for the CTR prediction problem. The increase shows the importance of online advertising in the last few years and how it is taking over traditional advertising. The second is the shift from interest in methods such as LR and FMs to ANN starting 2016 with five papers published that year followed by three in 2017 and two more as of June 2018. Therefore, ANNs are selected as the method of choice for CTR prediction.

Chapter 3

Methodology

The proposed methodology is a combination of an ML model and a metaheuristic. Suppose a dataset that comprises attributes of users such as the device used, the gender, age, and location, features of the ads such as the product marketed, type, and size as well as an outcome $y \in (0, 1)$ denoting a click event. Given this dataset, it is conceivable to determine the probability that a click occurs. Hence, the purpose of the ANN is to associate the given features to a likelihood of a click $\hat{y}$ occurring. On the other hand, the metaheuristic makes use of the ANN's predictions as a fitness function to optimise the ad's features. Preferably to a one-fits-all strategy, the methodology's focus is turned to each user independently. Leading to a considerable improvement in the marketing campaign's effectiveness measured by the CTR as well as the users' experience.

In that sense, this chapter is organised as follow: Section 3.1 covers the initial work on the data. Section 3.2 gives an overview of the methodology before diving into the details of the ANN in Section 3.3 and GA in Section 3.4.

3.1 Feature Engineering

Feature engineering is a complicated and time-consuming task that aims to convert a particular dataset to make it more fit for ML models. Regarding the dataset used, the first step is a selection of a representative 10%. The subset is then transformed and hashed before being oversampled to deal with the class imbalance.

3.1.1 Data Selection

For experimentation purposes, a 10% subset is selected from the original dataset. During the subset selection, it is imperative to keep the same distribution of positive and negative samples because this will impact the training stage. Thus, each observation in the dataset has a 10% chance of being selected and 90% of not being selected. The distribution is then verified to make sure both the original dataset and the subset have the same positive to negative observations ratio.

3.1.2 Feature Hashing

Features hashing is a fast and memory-efficient technique (Weinberger et al. 2009) that transforms all categorical and numerical features in the dataset to an integer ranging from 0 to D. The technique gives an

alternative to the use of associative arrays to link categorical data to numerical values (Goodrich & Tamassia 2008). Thus, a hashing function always outputs the same hash for a given value. In fact, the technique proved to be efficient with ML models such as SVM and LR. Especially when the data is too big to be loaded in memory (Li et al. 2011).

Feature hashing also solves the difficulty of converting categorical data to numerical values in an online learning context where the training is a continuous process. Moreover, it gives the possibility to extend the training of the model. Hence, feature hashing is an adequate technique for this methodology. It is necessary to select a hash algorithm that is both fast and avoids collisions (two different values leading to the same hashed value). Consequently, SHA1[1] is selected because of its output length of 160 bits (Eastlake 3rd & Jones 2001); which is enough to map the values in the dataset without collisions. The modulo D is fixed to 10^6 to reduce the large number generated by the hash function as suggested by Guo et al. who worked on the same dataset (Guo et al. 2017).

3.1.3 Synthetic Minority Over-sampling Technique

ML classifiers are usually evaluated based on their accuracy. However, when working on an imbalanced dataset, the accuracy is not an appropriate measure (Weinberger et al. 2009). The reason behind this is that the accuracy only captures one aspect when evaluating the performance of a classifier. Thus, when trained using an imbalanced dataset, the models maximise the accuracy by predicting the dominant class only (Provost 2000). Regarding the used dataset, there is a distinct imbalance between the positive and negative samples. Because of this, the training set is up-sampled using the Synthetic Minority Over-sampling Technique (SMOTE) (Weinberger et al. 2009).

Commonly, to deal with the imbalance, practitioners re-balance the dataset either by increasing the number of observations from the minority class (up-sampling) or decreasing the number of observations from the majority class (down-sampling). SMOTE falls within the category of up-sampling algorithms. Oppositely to more traditional up-sampling techniques that duplicate observations, SMOTE creates new synthetic observations. The algorithm works by finding a sample from the minority class and considers its K-NN in the features space. The difference between one of these neighbours and the points is then multiplied by a number ranging from 0 to 1. The new point is then added to the dataset (Weinberger et al. 2009).

Result: Over-sampled dataset
for *each point p in the minority class* **do**
    Find a k number of nearest neighbours to the point p;
    Randomly choose a number from these neighbours;
    Choose a random point between the original point p and the nearest neighbour;
    Add the point to the dataset with the minority class assigned;
end

Algorithm 1: SMOTE Algorithm Pseudo-code

---

[1] Secure Hash Algorithm 1

SMOTE is performed on the training set by using the R implementation in *UBL* (Branco et al. 2016). The distance metric to find the K-neighbours is set to the Overlap coefficient to take into consideration the nominal nature of the data and to avoid the creation of continuous values. The training set's imbalance is reduced to a 50:50 of positive to negative samples ratio. However, the test set is kept unchanged to simulate a real-life test for the model and to avoid predictions on synthetic data points that are meant for training only.

3.2 Methodology Overview

As specified earlier, the methodology is a combination of an ANN with a GA to optimise displayed ads on a user basis. The overall structure is presented in Figure 3.1.

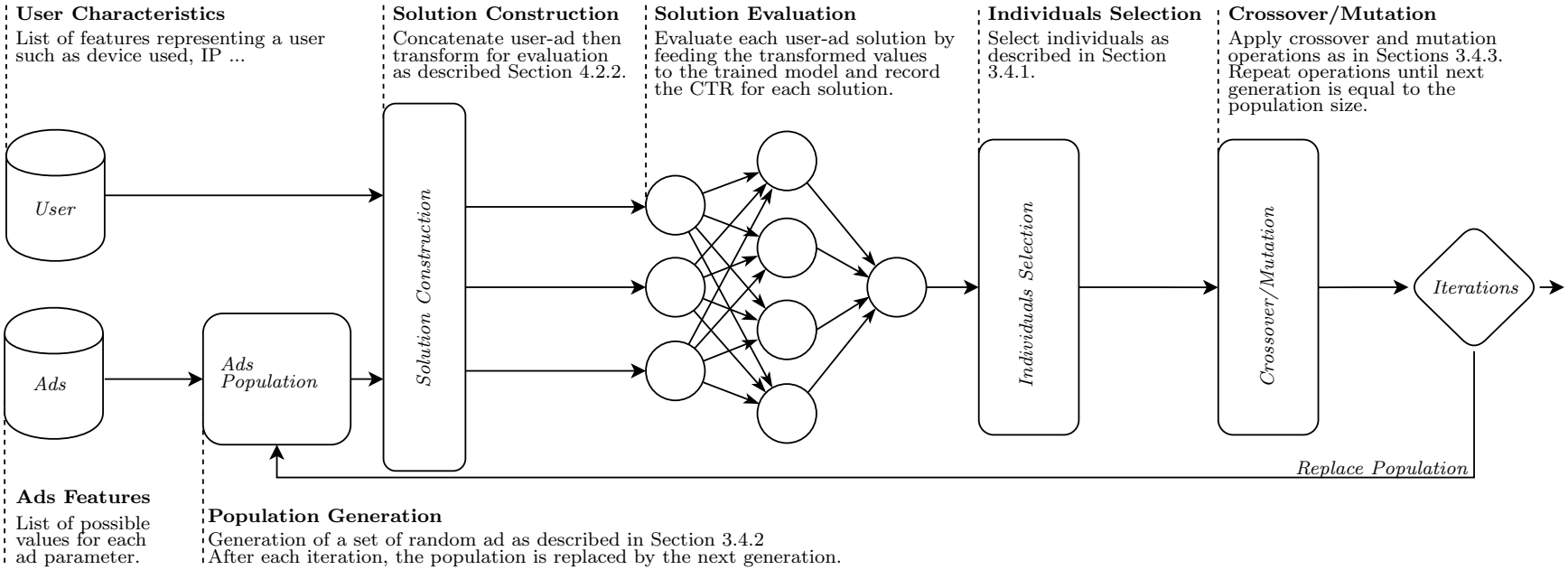

Figure 3.1: Methodology Structure

Figure 3.1 presents the interaction between the GA and the ANN after this last one is trained. While the GA drives the optimisation, the ANN takes part in the evaluation procedure where each combination of user and ads are given a predicted CTR. This CTR is then taken into consideration when selecting individuals for crossover and mutation operations. In that sense, Section 3.3 describes the ANN used while Section 3.4 covers the different parts of the GA.

3.3 Click-Through Rate Predictions with Artificial Neural Networks

The CTR prediction model is at the core of the methodology. It is therefore imperative to select a model that proved to be reliable for this kind of problems. Thus, the chosen model is an ANN and more precisely a Multi-layer Perceptron MLP with several hidden layers. ANN has proven to be effective as a CTR prediction method for different authors (Miralles-Pechuán et al. 2017, Chen et al. 2016, Covington et al. 2016, Qu et al. 2016, Chen et al. 2018, Jiang et al. 2018, Zhou et al. 2017). Similarly to these authors, the goal of the selected model is to predict the likelihood of a click occurring based on a set of features.

ANNs are a type of ML models that replicate the activity of neurons in a biological brain (Walczak 2018). A basic model contains l number of layers and n neurons. Each neuron receives input from the neurons in the previous layer and sends an output to the neurons in the next one until the output is reached. Figure 3.2 describes a basic one hidden layer MLP.

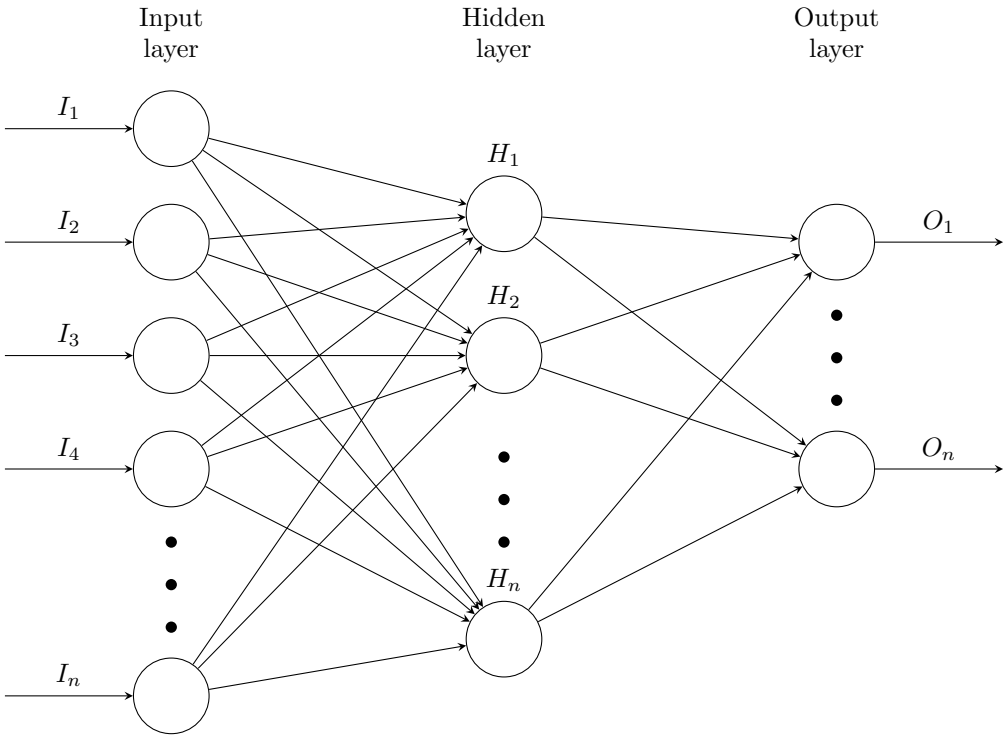

Figure 3.2: Artificial Neural Network Structure. Basic feedforward MLP with one input, one hidden, and one output layer. The input layer constitutes the entry of the data while the output layer gives the results. The arrows represent connections between the different layers and each one holds a weight that is adjusted during the training phase (Da Silva et al. 2017).

The model in Figure 3.2 is referred to as a Multi-Layer Perceptron. Each arrow represents an adjustable value called weight (ω) that allows the model to learn and better approximate the output (Walczak 2018). The inputs received by each node are multiplied by their corresponding weights (ω) and summed before being transformed using an activation function. The activation function is necessary to create a non-linear relationship between the inputs and outputs while keeping the values in a specific range (Da Silva et al.

2017). Figure 3.3 summarises the operations occurring at the neuron level.

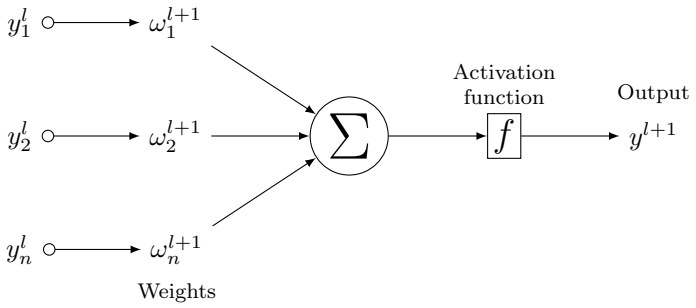

Figure 3.3: The Artificial Neuron. Representation of the operations occurring at the neuron level in an ANN.

The sum operation performed by the neuron is expressed in Equation 3.1. An example of an activation function, which is Sigmoid, is expressed in Equation 3.2 (Da Silva et al. 2017).

$$z_i^{l+1} = \sum_{i=1}^{n} \omega_i^{l+1} \cdot y_i^l \tag{3.1}$$

$$y_i^{l+1} = f(z_i^{l+1}) = \frac{1}{1 + exp(-z_i^{l+1})} \tag{3.2}$$

Thus, each node receives values y_i^l from the nodes in the previous layer l. These values are multiplied by their corresponding weight w_i^{l+1} and summed. The sum is then transformed by the activation function $f(x)$ before being sent to the next layer as y^{l+1} (Srivastava et al. 2014).

3.3.1 Backward Propagation

The essential element that allows ANNs to better approximate the outputs is the weights ω. These weights are calibrated through a process called Backward Propagation (Backpropagation) (Kriesel 2007) that combines two stages. The first stage is the forward propagation to estimate the predictions $\hat{y}$ for an input I. The outputs are then compared to the actual values y to assign an error value to the current model. The second stage is the backpropagation since the algorithm goes back to each node, assign an error value to it based on its contribution to the output, and adjust the weights accordingly to better approximate $\hat{y}$ in the next iteration and minimise the error value (Da Silva et al. 2017).

3.3.2 Evaluation Metrics

The AUC and Weighted F-Score are selected as metrics of choice to evaluate the hyperparameters of the model and compare it to other classifiers.

Area Under Receiver Operating Characteristic

The AUC or more precisely AUROC represents the area under the receiver operating characteristic (ROC) curve. While the ROC is a visual representation of the quality of a binary classifier, the AUC quantifies this by computing the integral of the ROC. It is a measure that has proved to be reliable for this type of problems through multiple studies (Aryafar et al. 2017) and is more appropriate for use when working with

an imbalanced dataset (Provost & Fawcett 2001, Chawla et al. 2002, Ling & Li 1998). Thus, if a model reacts negatively to the imbalanced dataset by predicting only one class, the AUC is negatively impacted.

Weighted F-Score

The second metric that is used is the Weighted F-Score. The F-Score is a measure of a classifier's accuracy that takes into consideration the precision and recall. More precisely, it is the harmonic mean of two metrics as shown in Equation 3.3 (Sasaki et al. 2007).

$$F - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3.3}$$

With the $Precision = TP/(TP + FP)$ and $Recall = TP/(TP + FN)$ (Chawla et al. 2002) and TP referring to True Positives, FP to False Positives and FN to False Negatives. The Weighted F-Score is a modified F-Score that accounts for class imbalance in a dataset. The measure calculates the F-Score for each label then returns the measure weighted by the number of instances in each class.

3.4 Optimization with Genetic Algorithms

A Genetic Algorithm (GA) is a bio-inspired metaheuristic used to find near-optimal solutions to optimisation problems by using a set of operators such as crossovers and mutations (Mitchell 1998). Its implementation is initialised with a population of random solutions that are evaluated using a fitness function. Based on the fitness, individuals are assigned a probability of selection for crossover and mutation operations. Thus, the better solutions have a higher probability of progressing their *genes* to the next generation. The initial set of the population is then replaced by the new individuals. A generic algorithm for the GA is presented in Algorithm 2 (Whitley 1994).

Result: Near-optimal solution
Randomly generate a population of size N;
while *Convergence or maximum iterations not reached* **do**
    Evaluate the population using the fitness function;
    Select the fittest individuals;
    Perform crossover operation;
    Perform mutation operation;
    Update population;
end

Algorithm 2: Generic Genetic Algorithm Pseudo-code

In that sense, the metaheuristic contains a fitness function and population selection as discussed in Section 3.4.1, population generation in Section 3.4.2 as well as crossover and mutation operations in Section 3.4.3.

3.4.1 Fitness Function and Population Selection

The fitness function measures the quality of a generated solution. Its design takes a crucial part in the optimisation process as it influences the course of the search (Kramer 2017). Because of that and since the goal of the methodology is to optimise the CTR, the ANN built for CTR prediction is used as a fitness function as shown in Equation 3.4.

$$Fitness(individual) = Prediction(individual) \tag{3.4}$$

Regarding the selection method, a roulette wheel selection is used to make sure that the best solutions have higher probabilities to progress to next generations. Roulette wheel selection assigns probabilities to each individual that are proportionate to their fitness values. The equation used to compute each probability is presented in Equation 3.5 (Kramer 2017).

$$P_{individual_i} = \frac{fitness(individual_i)}{\sum_{j=1}^{N} fitness(individual_j)} \tag{3.5}$$

By using this approach, it is ensured that the best solutions have higher probabilities of surviving and improving throughout the following generations.

3.4.2 Population Generation

Each array of features used to predict the CTR comprises a part that is specific to the advertisement and a part that is specific to the user. However, because the goal is to find the most adequate advertisements for a user, it is necessary to make the distinction between the section to optimise and the fixed section. Figure 3.4 presents the structure of each record.

$$\begin{bmatrix} ad_1 & ad_2 & \dots & ad_n & | & user_1 & user_2 & \dots & user_m \end{bmatrix}$$

Figure 3.4: Array Characteristics Structure

The individuals in the population contain the advertisements' features while the user's characteristics are stored in a different array. The two arrays are concatenated to calculate the fitness since it has to be related to a user. However, all the selection, mutation, and crossover are performed on the advertisements' features only. The population also plays a role in generating a set of possible and optimised ads for a user. Hence, after optimising, the algorithm outputs a set of ads of size N that are appropriate for the user.

To stay within the range of possible ads' parameters, a two-dimensional array of possible values is given to the algorithm before the generation. These parameters represent a list of constraints for each cell. The structure of the list of parameters is presented in Figure 3.5.

$$\begin{bmatrix} ad_{11} & ad_{12} & \dots & ad_n \\ ad_{21} & ad_{22} & \dots & ad_n \\ \vdots & \vdots & \vdots & \vdots \\ ad_{n1} & ad_{m2} & \dots & ad_n \end{bmatrix}$$

Figure 3.5: List of Parameters Structure

For each cell, a random integer between 0 and the length of possible values for that cell is generated. The cell corresponding to the generated index is taken as the starting block. The operation is repeated until all the cells are filled and a number of individuals in the population reached. Thus, the population generation algorithm is summarised in Algorithm 3.

Data: Population size & Parameters List
Result: Initial population
while *Population size not reached* **do**
  for *each cell in chromosome* **do**
    Generate random integer in range 0 to N Parameters;
    Insert corresponding parameter in cell;
  end
  Insert chromosome in population array;
  Increment population count;
end

Algorithm 3: Population Generation Algorithm

3.4.3 Crossover and Mutation Operations

The crossover operator combines two individuals (parents) to produce two offsprings containing a combination of characteristics in the parents. A single point crossover is selected as the method of choice for the crossover in the GA (Kramer 2017). The operation consists of randomly splitting the two parents. Two offsprings are then constructed by combining a part from the first parent and a part from the second (Kramer 2017). The operation is presented in Figure 3.6.

Parents

$$\begin{bmatrix} ad_1 & ad_2 & ad_3 & ad_4 & ad_5 \mid & ad_6 & ad_7 & ad_8 \end{bmatrix}$$

$$\begin{bmatrix} ad_1 & ad_2 & ad_3 & ad_4 & ad_5 \mid & ad_6 & ad_7 & ad_8 \end{bmatrix}$$

Offsprings

$$\begin{bmatrix} ad_1 & ad_2 & ad_3 & ad_4 & ad_5 \mid & ad_6 & ad_7 & ad_8 \end{bmatrix}$$

$$\begin{bmatrix} ad_1 & ad_2 & ad_3 & ad_4 & ad_5 \mid & ad_6 & ad_7 & ad_8 \end{bmatrix}$$

Figure 3.6: Single Point Crossover

On the other hand, the mutation operation is used to maintain the diversity in the chromosome and avoid being stuck in a local minimum. Based on a low probability, one or more cells in an individual are randomly modified (Kramer 2017). The mutation operation is presented in Figure 3.7.

$$\begin{bmatrix} ad_1 & ad_2 & ad_3 & O & ad_5 & | & ad_6 & ad_7 & ad_8 \end{bmatrix}$$
$$\begin{bmatrix} ad_1 & ad_2 & ad_3 & M & ad_5 & | & ad_6 & ad_7 & ad_8 \end{bmatrix}$$

Figure 3.7: Mutation Operation

Combined, the mutation and crossover operations ensure convergence without being stuck in a local minimum.

Chapter 4

Experimentation & Results

The following chapter presents the different experimentation phases. The experimentation environment is specified in Section 4.1. In Section 4.2, the dataset used is described. The parameters for the ANN and the GA are tuned in Section 4.3 and Section 4.4 respectively. Finally, three different experiments are conducted to test the performance of the methodology in Section 4.5.

4.1 Experimentation Environment

The hardware used consists of an ASUS R541U with an Inter(R) Core$^{(TM)}$ i7-7500U 2.70GH, 8 GB of RAM and an NVIDIA Geforce 920M. The methodology was developed on Python 3.6. ANN is implemented using the library *Keras* (Chollet et al. 2015). Prior to the experiment, the randomness is fixed to compare the configurations.

4.2 Data Description

The dataset used is a public dataset provided by Avazu[1] and contains more than 40 million records of CTR data of ads for mobile phone. Avazu is a global mobile advertising platform that targets users through games, apps, and mobile web pages in more than 130 countries. The platform gives its clients the possibility to target users based on location, time, device used, internet connection and so on (*Avazu Click Through Dataset* 2015). Avazu provided the dataset for a competition that was organised on Kaggle and contains 24 features of mostly categorical variables. These features contain information about the users such as the devices, app, site visited, as well as ads' parameters. However, ads' parameters were anonymised by being converted to categorical values by Avazu. The list of features is presented in Table 4.1.

[1]Dataset: https://www.kaggle.com/c/avazu-ctr-prediction/data

Feature Name	Data Type	Data Description
id	Numerical	ID of the ads displayed to the user.
click	Binary	Target Variable. 1 represents a click and 0 not a click.
hour	Date & Time	Date and time of the record in YYMMDDHH format.
banner_pos	Categorical	The position of the displayed ad.
site_id	Categorical	ID of visited website.
site_domain	Categorical	Domain of visited website.
site_category	Categorical	Category of visited website.
app_id	Categorical	ID of used app.
app_domain	Categorical	Domain of used app.
site_category	Categorical	Category of used app.
device_id	Categorical	Users' device ID.
device_ip	Categorical	Users' IP address.
device_model	Categorical	Users' device model.
device_type	Categorical	Users' device type.
device_conn_type	Categorical	Users' connection type.
C1 and C14 to C21	Categorical	Anonymous features representing ads parameters.

Table 4.1: Dataset Description. The table above showcases the set of features used for this experiment prior to feature engineering. The feature C1 and C14 to C21 are grouped and represent the ads parameters.

The dataset is imbalanced with only 17% of the records being clicks and 83% being non-clicks. For experimentation purposes, a sample of 10% of the original data is randomly selected while taking into consideration the data imbalance. Thus, the experiments are performed on a 4 million sample containing 17% of clicks only. The sample is split into training, validation, and testing sets based on a 50:25:25 split. In this case, a 25% validation and testing splits ensure a fair test and validation size after the training set is oversampled using SMOTE. The original, training, testing and validation datasets' distributions are displayed in Figure 4.1

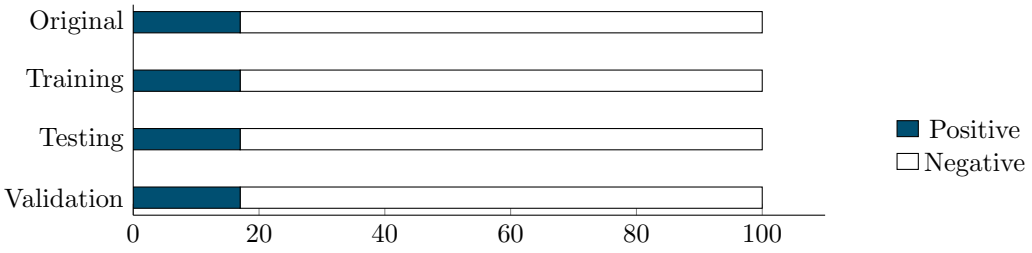

Figure 4.1: Dataset Composition. Distribution of the dataset in terms of clicks where the positive samples refer to clicks while the negative refer to non-clicks.

4.2.1 Synthetic Minority Over-Sampling Technique

After splitting the dataset, SMOTE is applied. As discussed in Section 3.1.3, SMOTE is applied to the training set only to avoid testing and validating on synthetic data points and to simulate a real situation of an imbalance. Thus, the distribution of the testing and validation sets is kept to 17% clicks and 83% non-clicks. The training set, on the other hand, is updated. The original size of the training set was 2 021 983. After performing SMOTE, the training set increased by 1 334 801 instances to reach a total of 3 356 784. Most importantly, the ratio of positive to negative observations changed from 17:83 to 50:50.

4.2.2 Feature Engineering

For the model to fully capture the effect of time and day of the week on the clicks, the feature *hour* is split into *time* and *date*. The original feature is represented as YYMMDDHH where the time is displayed on an hourly basis. Thus, it is separated into dummy variables representing the hour of the day from 00 to 23. The feature *date* is converted to values ranging from 0 to 6 to represent the day of the week. It is then converted to dummy variables. The transformation process is summarised in Figure 4.2.

Pre-Feature Engineering

$$[YYMMDDHH]$$

Post-Feature Engineering

$$[hour_0 \quad hour_1 \quad \dots \quad hour_{23} \quad day_0 \quad day_1 \quad day_2 \quad day_3 \quad \dots \quad day_6]$$

Figure 4.2: Feature *hour* transformation.

By applying this transformation, the effect of the day of the week and time of the day are taken into consideration when predicting where some days might have more clicks than some others.

4.2.3 Dataset Summary

To summarise the different processes applied to each split of the dataset, Figure 4.3 displays the splits, as well as the use of each dataset for the rest of the paper.

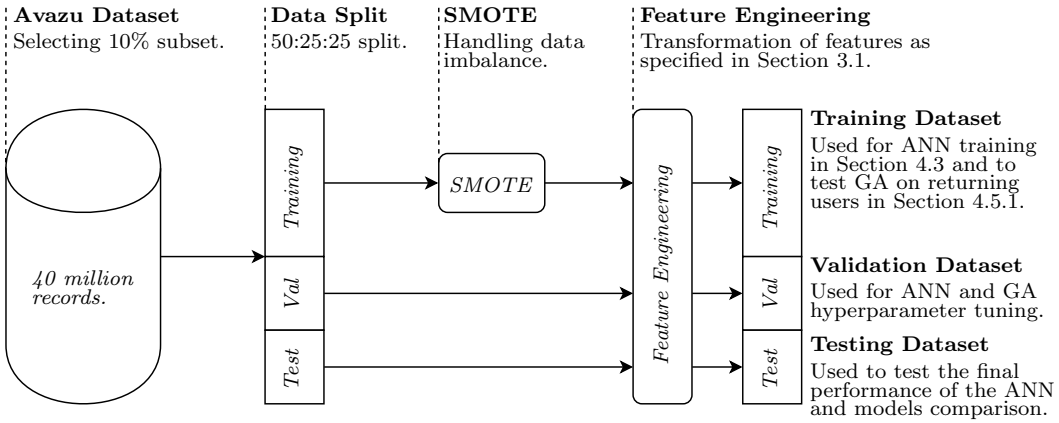

Figure 4.3: Dataset Handling Summary

After preparing the different datasets to be used, the next is to train the ANN using the training set and validate its results on the validation set.

4.3 Artificial Neural Networks Hyperparameter Tuning

The following section goes through the different hyperparameters and their effect on the AUC and weighted F-Score. The starting parameters are displayed in Table 4.2.

Parameter	Value
Hidden Layers Activation Function	ReLU
Output Layer Activation Function	Sigmoid
Optimizer	Adam
Learning Rate	0.00001
Nodes per layer	52
Number of Epoches	30

Table 4.2: Initial Model Parameters.

Hyperparameter tuning includes the depth of the network, number of neurons in each layer, network shape, activation functions, optimisers, dropout rates, and batch size. Once the optimal parameters fixed, a convergence analysis is performed before comparing the ANN's results to other models.

4.3.1 Network Depth

The depth of the network denotes the total number of hidden layers in the model. To determine this one, the model is tested for a number of hidden layers ranging from 2 to 7. The results in terms of AUC and Weighted F-Score are showcased in Figure 4.4.

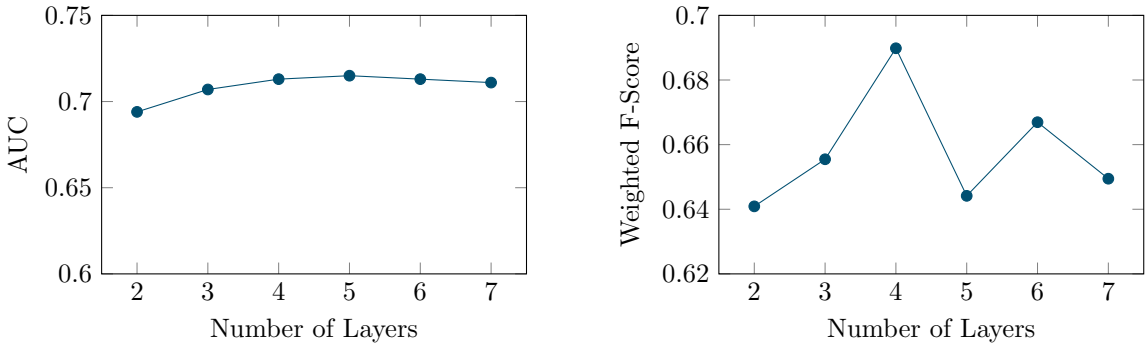

Figure 4.4: Model Depth Results. The AUC (left) and weighted F-Score (Right) resulting from testing increasing number of layers in the model.

Based on Figure 4.4, a model with a low number of layers is already able to predict the CTR with 0.69 AUC and 0.64 weighted F-Score. However, as the number of layers increases, the performance improves until the model reaches five layers where it starts slowly decreasing. Hence, the number of layers selected for the model is four as it gets the highest weighted F-Score of 0.689 and a high AUC of 0.713.

4.3.2 Number of Nodes per Layers

After determining the number of layers in the network, the subsequent step is to determine the number of nodes in each one. To achieve this, the four-layer model is tested with a number of nodes per layer corresponding to 0.5, 1, 2 and 4 times the dimension of the input. Thus, the number of nodes to be tested for each layer is 26, 52, 104 and 208. The results are displayed in Figure 4.5.

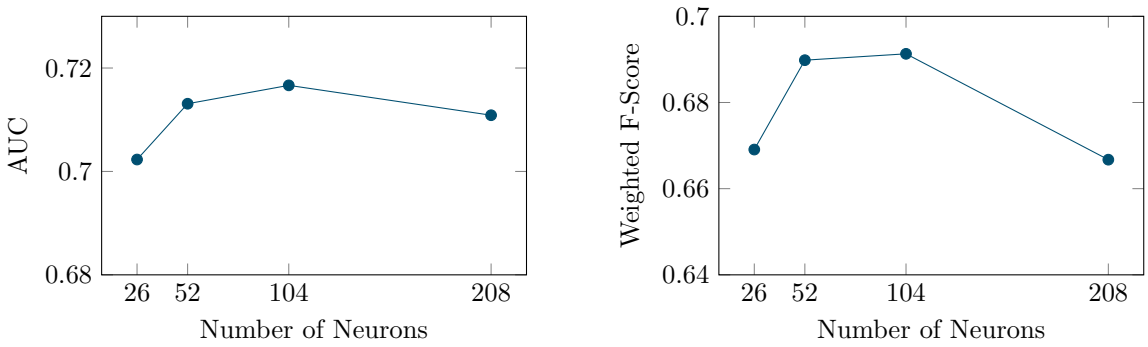

Figure 4.5: Model Nodes Results. The AUC (left) and weighted F-Score (right) resulting from different numbers of nodes in each layer.

Based on Figure 4.5, increasing the number of nodes has a slight but noticeable effect on both the AUC and weighted F-Score. However, after reaching a number of nodes equivalent to four times the input size, the performance starts decreasing. In that sense, the number of nodes selected for each layer is 104 with 0.717 AUC and 0.691 weighted F-Score.

4.3.3 Network Shape

The shape of the network represents the distribution of the nodes in each layer. The different shapes that are tested are constant, diamond, increasing and decreasing. The constant shape refers to a fixed number of nodes per layer. The decreasing and increasing shapes refer to a decreasing and increasing number of nodes per layer respectively. The diamond shape refers to an increasing number of nodes until the centre of the network and a decreasing number of nodes from the centre to the output layer. Table 4.3 showcases the shapes as well as the number of nodes per layer in each one.

Shape	**Layer 1**	**Layer 2**	**Layer 3**	**Layer 4**
Constant	104	104	104	104
Decreasing	200	136	72	8
Increasing	8	72	136	200
Diamond	52	156	156	52

Table 4.3: Network Shapes Composition. Number of nodes in each layer for each network shape.

The AUC and weighted F-Score resulting from the experiment are displayed in Figure 4.6.

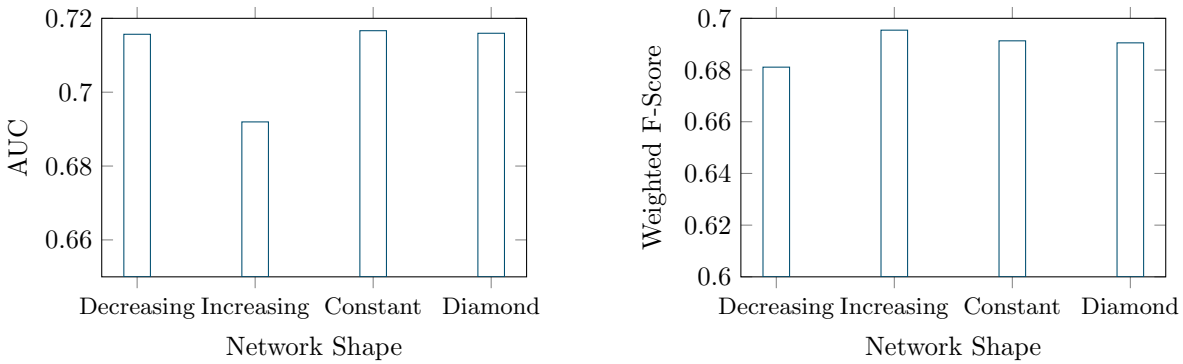

Figure 4.6: Model Shapes Results. The AUC (left) and weighted F-Score (right) resulting from testing the different shapes presented in Table 4.3.

Based on Figure 4.6, a constant number of nodes per layer seems to perform slightly better than the other three configurations in terms of AUC. These findings validate the conclusions of Larochelle et al. who claim that having a constant width model performs better than other arrangements (Larochelle et al. 2009). Consequently, the model's shape is fixed to a constant one with 104 nodes in each one of the four layers.

4.3.4 Activation Function

The role of the activation is to create a non-linear relationship between the inputs and the outputs by using a non-linear activation function while keeping the values within a specified range. For instance, a Sigmoid function acts like an LR by keeping all the values in a (0, 1) range (Da Silva et al. 2017). For this reason, Sigmoid is the activation function for the output layer. However, for the hidden layers, different activation functions are tested. These functions are ReLU, Leaky ReLU, as well as Tanh. To display the properties of each one, the functions are displayed in Figure 4.7.

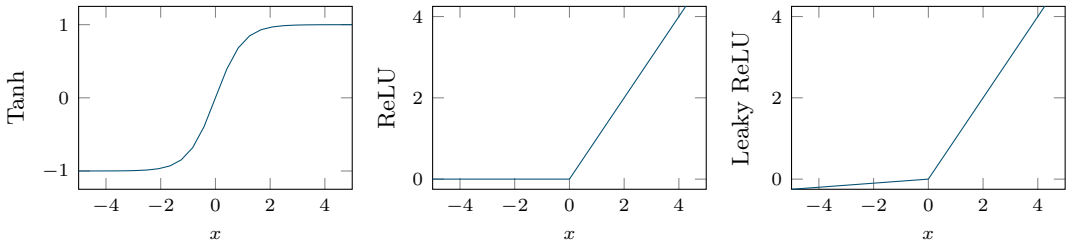

Figure 4.7: Activation Functions. The set of tested activation functions include the the hyperbolic tangent $tanh(x)$ (Maas et al. 2013) (left), Rectified Linear Unit $ReLU(x)$ (Nair & Hinton 2010) (middle) and the Leaky Rectified Linear Unit $LeakyReLU(x)$ (Maas et al. 2013) (right).

The results in terms of AUC and weighted F-Score are reported in Figure 4.8.

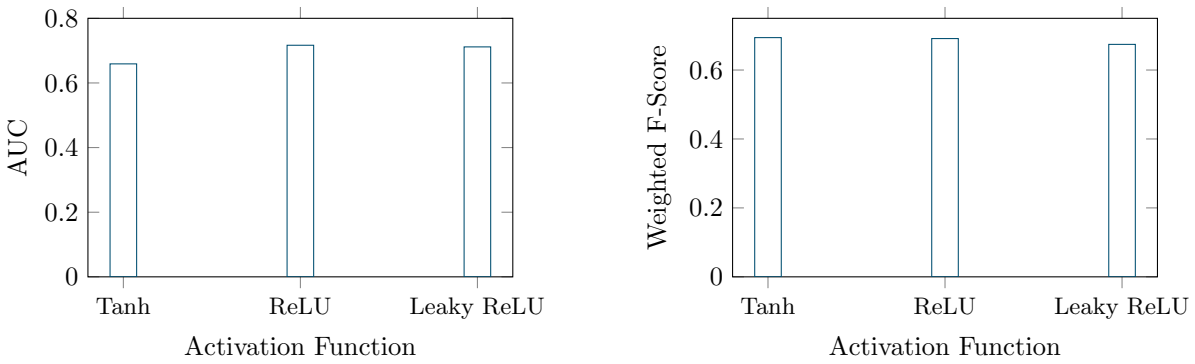

Figure 4.8: Activation Functions Results. The AUC (left) and weighted F-Score (right) resulting from testing the activation function presented in Figure 4.7.

Based on Figure 4.8, the optimal activation function for the hidden layer is ReLU with an AUC of 0.717 and a weighted F-Score of 0.691. However, the distinction between ReLU and Leaky ReLU is minimal suggesting that the use of one of the two function is good. The chosen activation function for the model is ReLU.

4.3.5 Optimiser

Different optimisers are tested to find the most appropriate one for the dataset used. In this sense, the optimisers tested are Adam, Nadam, SGD with 0.9 momentum and SGD with Nesterov and 0.9 Momentum. The results in terms of AUC and weighted F-Score are displayed in Figure 4.9.

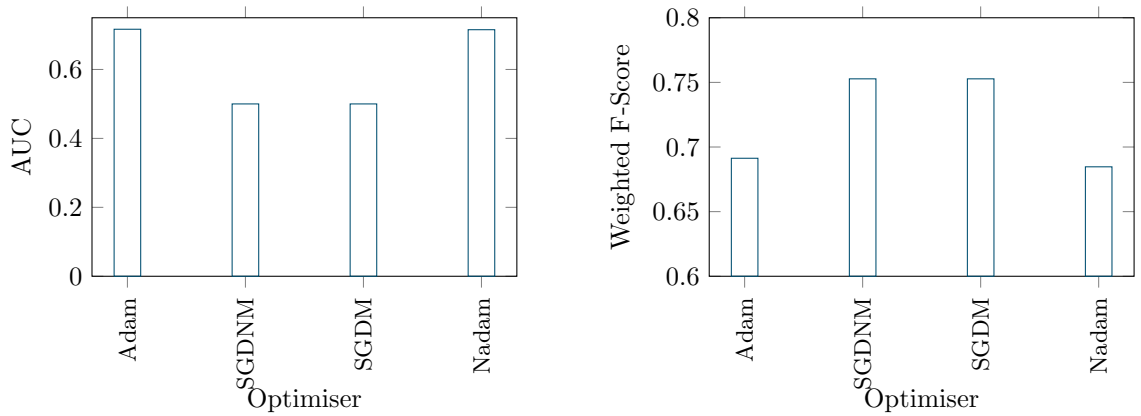

Figure 4.9: Optimisers Results. The AUC (left) and weighted F-Score (right) resulting from testing various optimisers. SGDNM refers to the use of SGD with Nesterov and 0.9 Momentum. SGDM refers to the use of SGD with 0.9 Momentum.

Based on Figure 4.9, the results of the tested Adam and Nadam are close. However, Adam gives slightly better results of 0.717 AUC and 0.691 F-Score. Regarding the other optimisers, having an AUC of 0.5 and a high F-Score, in this case, indicates overfitting of the model towards the majority class. To demonstrate this overfitting, Figure 4.10 is a confusion matrix of SGDMN and SGDM.

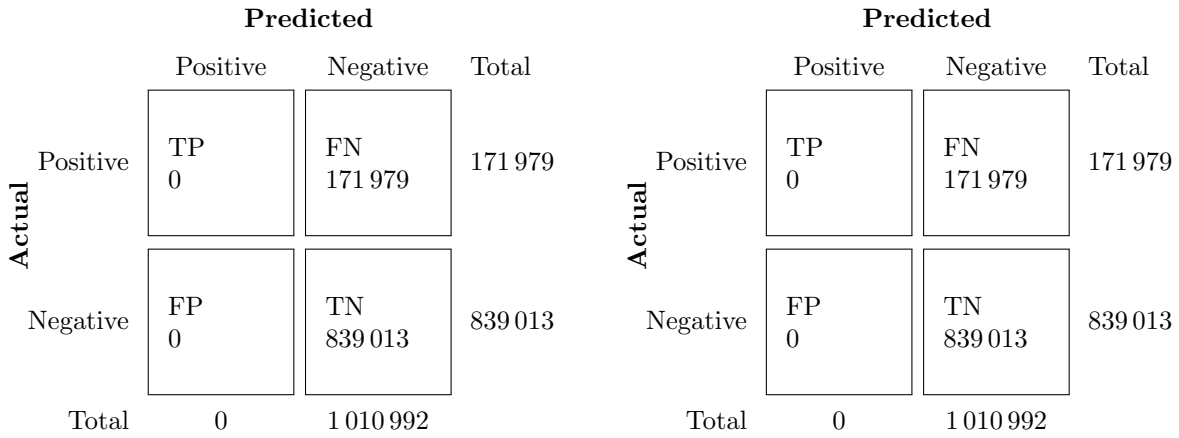

Figure 4.10: Optimisers Confusion Matrix. Confusion matrix for SGDMN (left) and SGDM (right). The matrices show that the two models trained with SGDMN and SGDM only predict the negative class which is the non-clicks.

Figure 4.10 shows the overfitting of the two models where they only predict the majority class. Thus, in this case, having a high weighted F-Score combined with a low AUC shows that the model is overfitting the majority class. In that sense, the optimiser used is Adam with a learning rate of 0.00001.

4.3.6 Dropout Rate

The dropout rate refers to the probability that a node from a layer will be removed to tackle overfitting. In other words, it is a regularisation technique in ANNs that adds noise to the hidden units to prevent it from adapting too much to the training set (Srivastava et al. 2014). To examine if adding dropout is suitable to the model, a dropout is added after each hidden layer with a rate ranging from 0 to 0.4. The results are showcased in Figure 4.11.

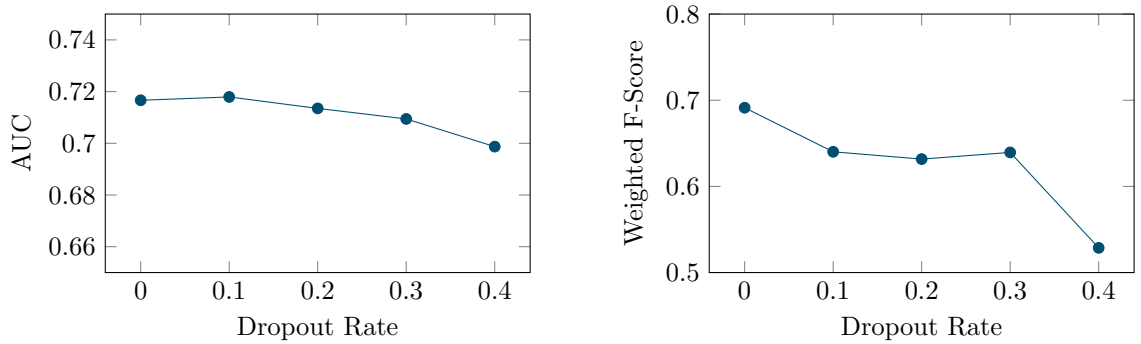

Figure 4.11: Dropout Rates Results. The AUC (left) and weighted F-Score (right) resulting from testing an increasing dropout rate in each layer.

Figure 4.11 shows that the model reacts negatively to the addition of a dropout rate. This is revealed by a drop in both AUC and weighted F-Score when introducing dropouts. This drop worsens as the dropout rate is increased. Thus, the dropout technique is omitted for the final model.

4.3.7 Batch Size

The batch size refers to the number of observations that are fed to the model at once during the learning phase. Having a low batch size doesn't allow the model to correctly see the big picture of the data while a large batch size makes the training too fast. Thus, to find the optimal batch size, the training is tested with 256, 512, 1024 and 2048. Figure 4.12 showcases the results in terms of AUC and Weighted F-Score.

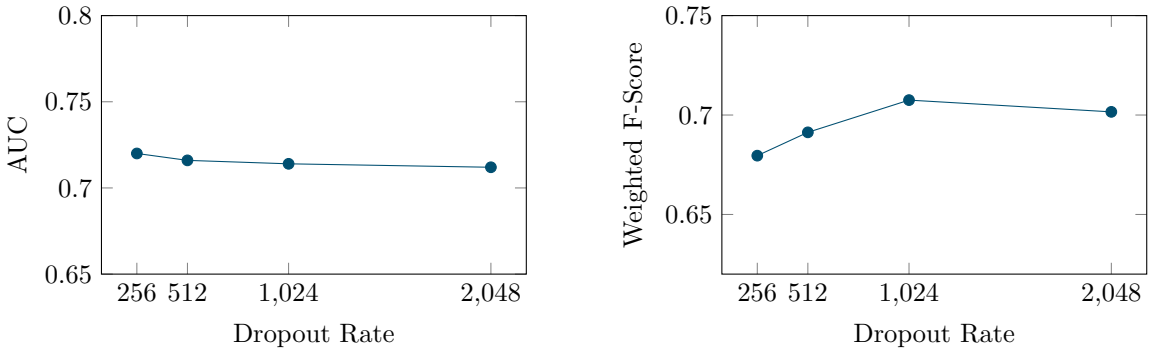

Figure 4.12: Batch Size Results. The AUC (left) and weighted F-Score (right) resulting from testing different batch sizes during the training.

A batch size of 256 does not seem to provide the model with enough data during the training. As the batch size is increased, the model performs better in terms of weighted F-Score that reaches 0.71 when the batch size is 1024 without having a significant impact on the AUC. However, when the batch size reaches 2048, the performance starts decreasing. Thus, the optimal batch size selected is 1024.

4.3.8 Convergence Analysis

The number of epochs is set to 300 to find the convergence point. The AUC is then recorded after each epoch. Figure 4.13 shows the evolution of the AUC.

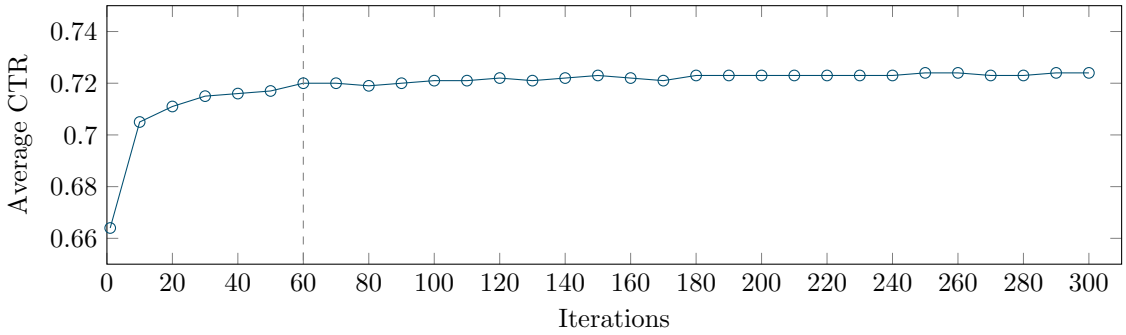

Figure 4.13: ANN Convergence Analysis. The performance of the model on the validation set in terms of AUC after each epoch. The model starts converging after 60 epochs.

Based on Figure 4.13, there is a significant increase in the AUC from 0 to 20 epochs. After reaching 20 iterations, the AUC keeps slightly increasing until reaching 60 epochs where the increase after that is not significant. In fact, the increase from the first epoch to epoch 60 is from 0.664 to 0.72 while the increase

from epoch 60 to 300 is from 0.72 to 0.724. In that sense, the optimal number of epochs selected to train the model is fixed to 60 epochs.

4.3.9 Predictive Models Comparison

To confirm the findings from the literature review claiming that ANN performs better than other state-of-the-art techniques for CTR predictions, the developed model is compared to various models. In this case, the testing set is used to test the performance instead of the validation set while the training set remains unchanged. These models are LR, NB, NB and DT. The results in terms of AUC and weighted F-Score for each model are reported in Figure 4.14.

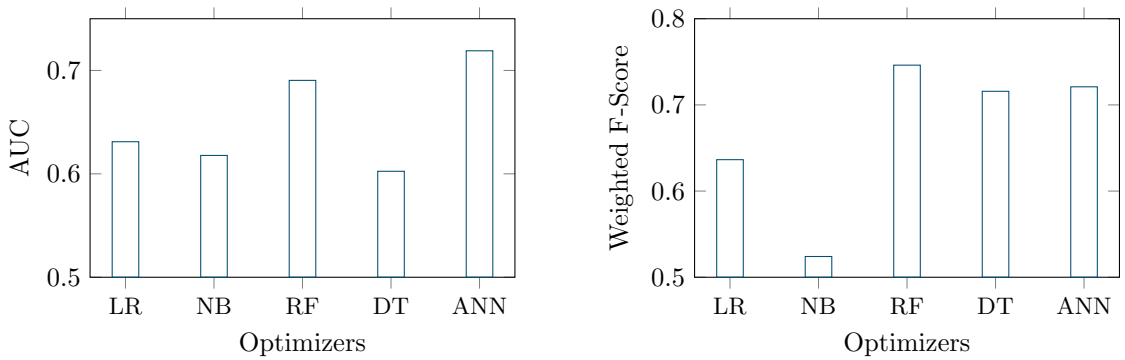

Figure 4.14: Models Comparison Results. The AUC (left) and weighted F-Score (right) for Logistic Regression (LR), Naïve Bayes (NB), Random Forest (RF), and Decision Trees (DT). All the models were trained using the same dataset and tested on the testing set.

As reported by Figure 4.14, the ANN performs significantly better than all the models in terms of AUC. Regarding the weighted F-Score, some models predict more negative classes which leads to higher recall and precision rates but causes a decrease in predicting the positive class as reported by the AUC. Thus, the final performance of the ANN on the testing set is 0.719 AUC, 0.721 weighted F-Score, and 0.686 accuracy. The ANN also performs better than a combination of ANN and LR suggested by Chen et al. who also used a sample of the same dataset and reached an AUC of 0.7127 (Chen et al. 2017).

4.4 Genetic Algorithms Hyperparameter Tuning

After training the ANN, it is introduced as a fitness function for the GA. The hyperparameters for the GA itself are then tuned. These hyperparameters are the crossover and mutation rates, the population size, and the number of iterations.

4.4.1 Crossover and Mutation Rates

The two main parameters to be tuned are the crossover and mutation rates. While the crossover rate ensures that new solutions are created in each generation, the mutation rate guarantees that the optimisation is not stuck in a local minimum (Kramer 2017). In that sense, it is crucial to have a high crossover rate that leads to the creation of a new generation after each iteration. The opposite applies to the mutation rate that needs to be low enough so that the search does not become a random one. Therefore, the values tested for each parameter range from 0.025 to 0.20 for the mutation rate and 0.5 to 0.9 for the crossover rate. Table 4.4 represents the solutions of the optimisation given by the combination of each mutation and crossover rate.

Rate	0.5	0.6	0.7	0.8	0.9
0.025	0.88	0.88	0.89	0.92	0.89
0.05	0.8	0.82	0.83	0.85	0.85
0.075	0.75	0.78	0.78	0.78	0.82
0.1	0.69	0.7	0.73	0.72	0.75
0.125	0.65	0.66	0.71	0.72	0.68
0.15	0.6	0.62	0.62	0.65	0.67
0.175	0.56	0.58	0.6	0.61	0.64
0.2	0.54	0.56	0.56	0.57	0.6

Table 4.4: Mutation and Crossover Rates Tuning. The average CTR achieved for 5 000 users using a combination of different mutation (rows) and crossover (columns) rates. The starting average CTR for the 5 000 users is 0.36 and the table presents the average CTR after 50 generations with a population size of 50. The best results are achieved by low mutation and high crossover rates.

Based on Table 4.4, it is apparent that a combination of high crossover and low mutation rates lead to better solutions. Thus, the selected values are 0.025 for the mutation rate and 0.8 for the crossover rate.

4.4.2 Population Size

The population size represents the number of solutions throughout the generations in the GA. In this case, it is the number of randomly generated ads at the beginning of the algorithm that are evaluated using the ANN and that will be used later on for mutation and crossover operations before being replaced by the next generation of the same size. Setting the right population size is a crucial step because a small population does not allow the algorithm to fully capture different facets of the data. A substantial population size, on the other hand, covers too much of the solution space and does not allow the algorithm to quickly converge into a good solution. For this reason, the population size is tested for values from 10 to 100 with a 5 increment. The results in terms of optimised CTR are displayed in Figure 4.15

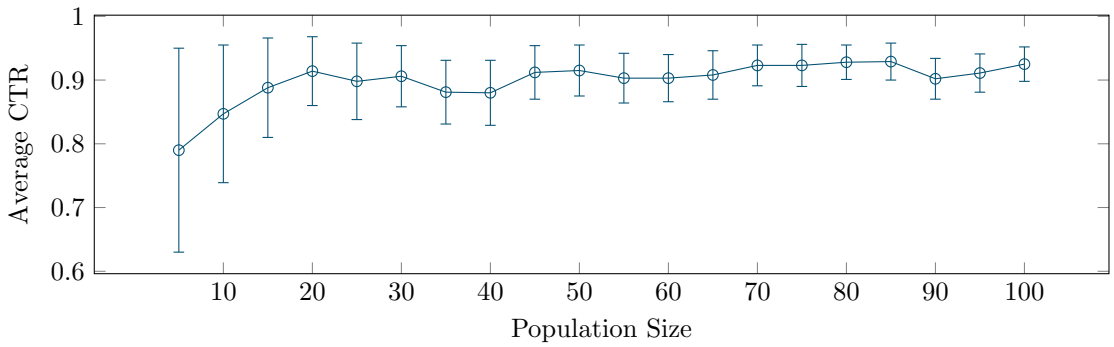

Figure 4.15: Population Size Tuning. The average CTR and standard deviation after optimising the ads for 5 000 users using different population sizes. The mutation and corossver rates used are 0.025 and 0.8 respectively. The number of iterations is set to 50.

As expected, A small population size does not allow the algorithm to reach convergence. Thus, the worst results are given by population sizes of 10 and 20 who only slightly improved the average CTR for the solutions. As the number of population increases, the optimisation results get better until they start converging at 70. In that sense, the population size is updated from the initial 50 to 70.

4.4.3 Convergence Analysis

In the context of GA, the convergence analysis refers to the number of generations needed until the algorithm reaches a good and stable solution. The GA with the 0.8 mutation and 0.025 crossover rates is analysed by setting the number of iterations to 1000 and recording the average CTR for each user after each iteration. The optimisation deals with a list of 5 000 users from the testing dataset. The results in terms of average CTR are presented in Figure 4.16.

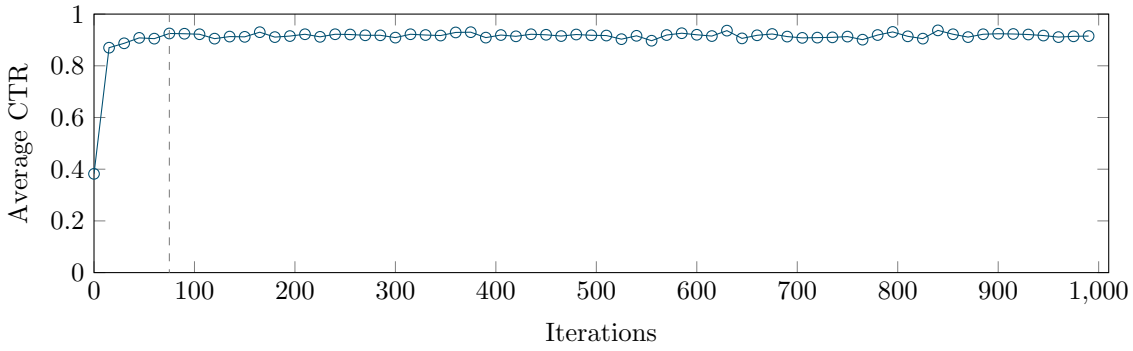

Figure 4.16: Genetics Algorithm Convergence Analysis. The average CTR recorded after each iteration for 5 000 users. The algorithm starts converting after 75 iterations.

From Figure 4.16, it is possible to see the start of the iterations where the average CTR is still low. However, as the algorithm creates new generations, the average CTR keeps increasing until reaching convergence around 75 iterations which is early for a GA. This is explained by the size of the search space resulting from a low number of possible configurations. However, as the number of parameters increases, it might be needed to increase the number of iterations to reach convergence. In that sense, the number of generations set for the rest of the experiment is 75.

4.5 Methodology Performance Testing

Three different experiments are conducted to test the performance of the methodology. The first experiment in Section 4.5.1 deals with the optimisation of returning users. A set of 50 000 users is selected from the training set to achieve this. The second experiment in Section 4.5.2 explores how the methodology handles new users. For this, a set of 50 000 users is extracted from the original 40 million rows dataset. Only users with a combination of characteristics that were not previously seen in the training set are selected. The last experiment in Section 4.5.3 aims at testing the optimisation under different constraints. For this, 50 000 mixed users are optimised on 50% of the ads' characteristics only.

4.5.1 Experiment 1: Returning Users

A subset of 50 000 is selected from the training set to showcase the methodology's performance. The users are divided into five batches containing 10 000 users each. For each user, the information available is the site or app used, the device, IP, connection type, and the date and time of the visit. From the testing set, advertisement features are extracted to be used as constraints during the optimisation. Thus, the goal is to find the best features for each user in each batch. The optimisation then runs with a 0.025 mutation rate, 0.8 crossover rate, 70 population size, and 75 iterations. The results are displayed in Table 4.5.

Batch	AVG Start	STD Start	AVG End	STD End	Min	Max	Time in ms
Batch 1	0.314	0.061	0.915	0.034	0.709	0.994	0.333
Batch 2	0.314	0.06	0.916	0.034	0.742	0.996	0.332
Batch 3	0.313	0.06	0.915	0.034	0.736	0.994	0.421
Batch 4	0.315	0.061	0.915	0.033	0.72	0.998	0.42
Batch 5	0.315	0.061	0.915	0.034	0.74	0.995	0.465

Table 4.5: Returning Users - Performance. The average CTR after optimising 50 000 returning users. AVG Start is the average CTR for the users prior to the optimisation while the AVG End is after the optimisation. The Min and Max represent the maximum CTRs for the users. Time in ms is the average time per user to optimise.

Table 4.5 shows the performance of the methodology in a normal context without any constraints. Thus, as expected, when ads are given randomly, the probability of a click occurring is low.

However, after 75 iterations of GA, the average CTR increases to 91.5% on average for the 50 000 users. Thus, the optimisation is able to find the right ads to show the right user to maximise the CTR. The minimum CTR found for a specific user is varying from 70.9% to 74.2% depending on the batch. This show that, even after the optimisation, some users have relatively lower probabilities of click. On the other hand, some users have extremely high CTR with the average maximum over the batches being not less than 99.8%.

To better investigate the predicted CTRs by the ANN before and after the optimisation, they are plotted in a histogram with a density function. This shows the distribution of the predicted CTRs before and after the optimisation. The results are showcased in Figure 4.17.

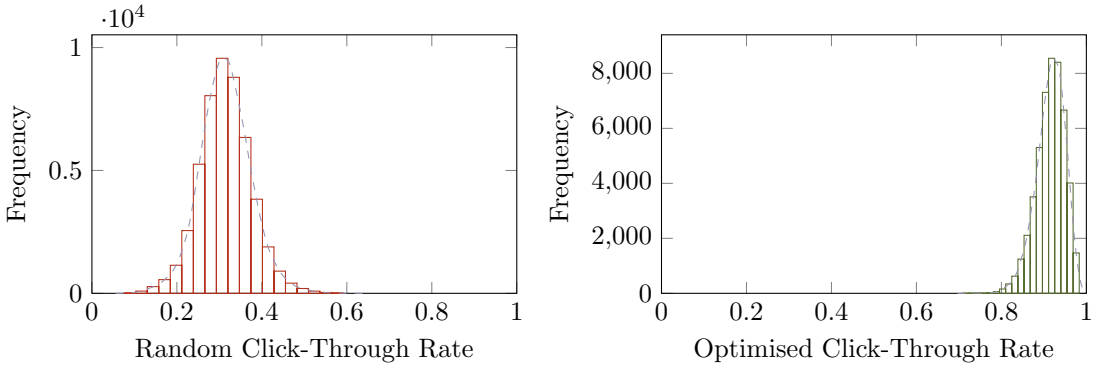

Figure 4.17: Returning Users - CTR Distribution. Histogram of the predicted CTR before (left) and after (right) the optimisation. The values represented are the average CTR for each of the 50 000 users in the five batches.

Figure 4.17 shows the distribution of the predicted probabilities before and after the GA optimisation. Before the optimisation, the predicted values for the CTR are concentrated between 0.2 and 0.4 with the maximum value being lower than 0.6. After the optimisation, the predicted CTR increases for most users. The main thing that Figure 4.17 shows is how the algorithm handles each user independently. Overall, the algorithm seems to handle all types of users since there is a moderate number of probabilities that are below 0.8. Regarding the advertisements created, each user has been assigned a unique best ad for a total of 50 000 unique ads created by the methodology.

4.5.2 Experiment 2: New Users

Similarly to the first experiment, a set of 50 000 users are divided into five batches and optimised. However, in this experiment, the users are filtered to only include the ones that are not found in the training set. In that sense, a new user refers to a new sequence of features that are not present in the training set but are present in the overall 40 million samples dataset. By doing so, the methodology is tested on the original dataset even though it was trained on a sample only. The results in terms of CTR are displayed in Table 4.6.

Batch	AVG Start	STD Start	AVG End	STD End	Min	Max	Time in ms
Batch 1	0.314	0.062	0.915	0.034	0.748	0.997	0.362
Batch 2	0.314	0.062	0.915	0.034	0.736	0.997	0.383
Batch 3	0.315	0.063	0.914	0.034	0.727	0.994	0.424
Batch 4	0.314	0.062	0.915	0.034	0.716	1.0	0.452
Batch 5	0.315	0.062	0.915	0.034	0.743	0.995	0.488

Table 4.6: New Users - Performance. The average CTR after optimising 50 000 new users. AVG Start is the average CTR for the users prior to the optimisation while the AVG End is after the optimisation. The Min and Max represent the maximum CTRs for the users. Time in ms is the average time per user to optimise.

Table 4.6 showcases the performance of the methodology on a set of users that were not seen in the training set. As for experiment 1, the methodology significantly improves the average CTR of the users in each batch. In fact, the results given for the new users are as good or better than the sample extracted from the previous

set. This provides an insight into the prediction power of the ANN used. Indeed, the ANN seems to focus on the interactions between the ads and the users instead of the relationship between the users' features themselves. Hence, when given a dataset containing old features formed in new combinations, the ANN is still able to evaluate the user-ad matching; which is then used by the GA to optimise the ads. To confirm these findings, the distribution of the predictions is displayed in Figure 4.18.

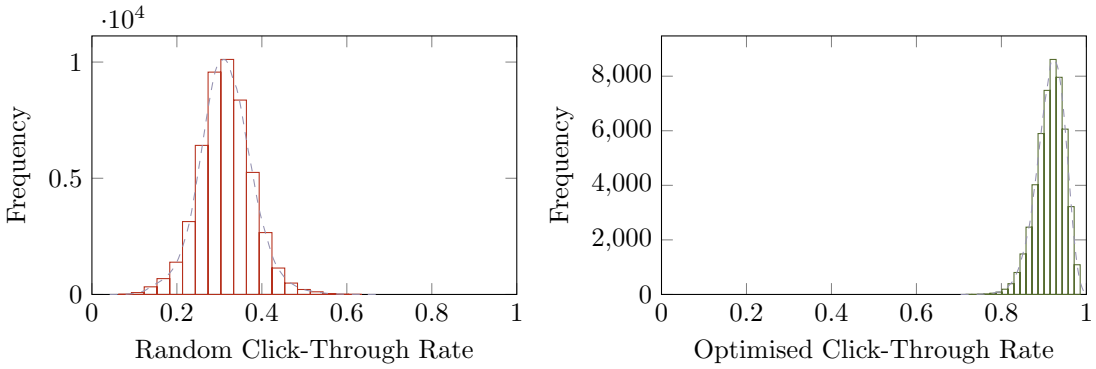

Figure 4.18: New Users - CTR Distribution. Histogram of the predicted CTR before (left) and after (right) the optimisation. The values represented are the average CTR for each of the 50 000 users in the five batches.

As expected, the distribution of CTRs for the new users is more concentrated around 90% CTR without being affected by the set of new users. In that sense, experiment 2 confirms that the methodology handles the arrival of new users and find adequate ads. Regarding the starting CTRs, the values are very similar to experiment 1. Once again, the number of unique best ads for each user is equal to the number of users; which is 50 000.

4.5.3 Experiment 3: Constraints Handling

Lastly, the methodology is tested for its ability to handle additional constraints on ads' features. To do so, and for each batch, 50% of the ads' features selected randomly are fixed to some specific values that the methodology is not allowed to change. Thus, the experiment will give an overview of how the methodology adapts to a small number of update-able features to optimise the CTR. Table 4.7 displays the features fixed for each batch.

Batch	**Constrained Features**
Batch 1	C1 - C15 - C17 - C19 - C20
Batch 2	banner_pos - C14 - C16 - C17 - C19
Batch 3	C15 - C16 - C18 - C20 - C21
Batch 4	C14 - C17 - C18 - C19 - C20
Batch 5	C1 - banner_pos - C14 - C16 - C21

Table 4.7: Ads Constraints - Parameters. The features that are fixed at the beginning of the optimisation for each batch; Each feature is fixed to one value randomly selected from the set of possible values.

The same procedure is followed for the optimisation but optimising only 50% of the features. The results in terms of CTR are displayed in Table 4.8.

Batch	AVG Start	STD Start	AVG End	STD End	Min	Max	Time in ms
Batch 1	0.21	0.103	0.91	0.052	0.154	0.998	0.45
Batch 2	0.25	0.077	0.836	0.073	0.434	0.984	0.539
Batch 3	0.28	0.121	0.904	0.039	0.642	0.998	0.414
Batch 4	0.499	0.123	0.963	0.024	0.677	1.0	0.378
Batch 5	0.175	0.074	0.897	0.059	0.557	0.998	0.382

Table 4.8: Ads Constraints - Performance. The average CTR after optimising 50 000 users with ads constraints. AVG Start is the average CTR for the users prior to the optimisation while the AVG End is after the optimisation. The Min and Max represent the maximum CTRs for the users. Time in ms is the average time per user to optimise.

Table 4.8 shows the performance on the batches with different constraints. Oppositely to the first two experiments, the CTR varies for each batch depending on the constraints set. The highest CTR attained is on batch four who achieved better results than the unconstrained optimisation. However, for this batch, even the starting CTR is relatively higher with 0.499; which shows that some of the constraints selected prior to the optimisation are already adequate for a large number of users.

However, for the other batches, the performance is slightly lower than the unconstrained optimisation with 0.904, 0.897, and 0.836 for batches three, five, and two respectively. The resulting standard deviation is also higher than the unconstrained optimisation. The distribution of predicted CTR is showcased in Figure 4.19.

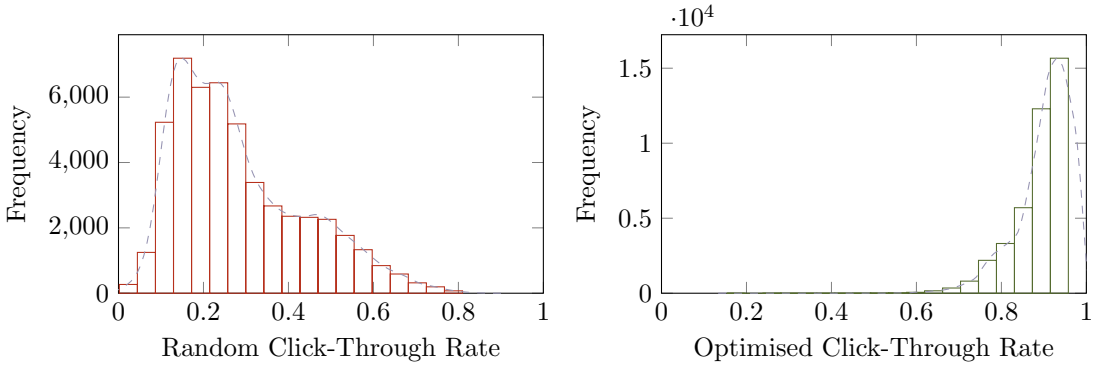

Figure 4.19: Ads Constraints - Distribution. Histogram of the predicted CTR before (left) and after (right) the optimisation with constraints. The values represented are the average CTR for each of the 50 000 users in the five batches.

Figure 4.19 confirms the findings from Table 4.8 where the values are more distributed than in experiments 1 and 2. However, even with the constraints, the methodology updates the unconstrained variables to accommodate to each user and achieve high CTRs.

Optimising with different constraints signals two critical factors to take into consideration. The first factor is that the features have different importance during the optimisation. This is shown by the difference in CTR resulting from different constrained features. The second factor is the discrepancy in the optimisation for the different users where some users get low results while others get high results. This shows that fixing some features might be beneficial for a type of users. These two findings lead to additional benefits of the methodology that are discussed in Chapter 5.

4.6 Experimentation & Results Summary

To summarise the findings, Section 4.3 tested different parameters for the ANN predictor that varied from the number of layers to the batch size. The final findings pointed to a four-layer model with 52 nodes per layer and with a combination of ReLU and Sigmoid as activation functions. The optimiser used is Adam with a learning rate of 0.00001 and did not include dropout layers. The training of the model was completed with a batch size of 1 024 over 60 epochs. The final AUC, weighted F-Score, and accuracy on the testing set are respectively 0.719, 0.721, and 0.686. The trained model was used as a fitness function for the GA with a mutation rate of 0.025 and a crossover rate of 0.8. The population size is set to 70 and the optimisation generated 75 generations before converging. After fixing the different parameters, three experiments were conducted to test the methodology's performance on returning and new users as well as with ads constraints. The results for the three experiments are summarised in Figure 4.20.

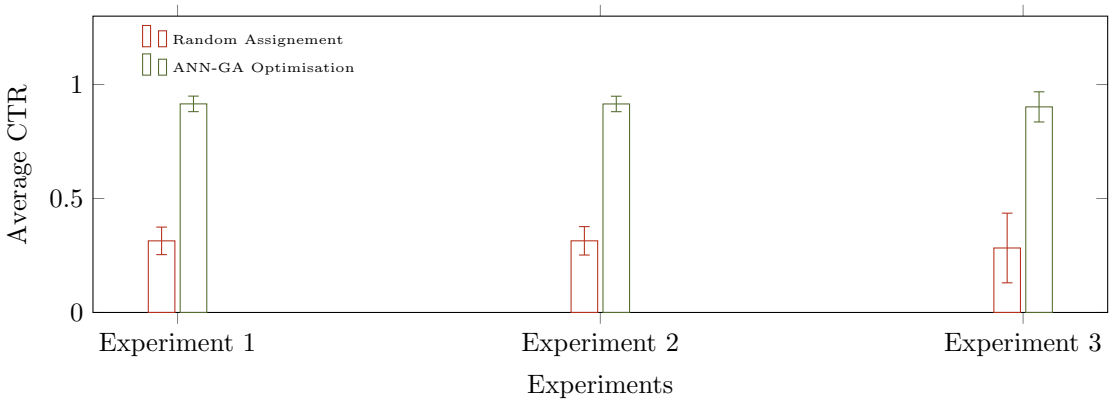

Figure 4.20: Experimentations Summary. Summary of the three conducted experiments in terms of CTR and standard deviation before and after the optimisation.

Experiment 1 showcases the performance of the methodology on returning users without constraints. Experiment 2 confirms the methodology's performance and ability to handle new users that were selected from the original 40 million samples dataset.

The findings from experiment 3 shed light on a specific use for the optimisation. Because the results included high CTR for some users and low CTR for some others, the methodology seems to be able to find the best users for some specific ads and gives more importance to some features compared to other. Thus, in addition to its regular use, the methodology can help find an optimal audience for a set of given features and distinguish the most important features for a specific audience.

Chapter 5

Implications for Research and Practice

The following chapter covers the implication of the proposed methodology from a practice and research points of view. The implications for practice covers the usage of the methodology from a business point of view in various contexts. The research implications focus on proposed research that follows the same path of the methodology.

5.1 Implications for Practice

From a practice point of view, the goal of the research conducted is to give advertisers a dynamic method to create marketing campaigns. In this sense, advertisers develop dynamic marketing campaigns that understand and adapt to the characteristics of each user. However, because of its ability to match users to ads, the methodology can be used for different other goals. Thus, the methodology can be used like the experiments in this paper to (i) find the best match of a user-ad with different ad parameter to choose from in a real-time basis. For advertisers with a more limited budget, the methodology can (ii) find the best audience for a specific ad or a set of limited ad parameter or (iii) find the best ads from a set of parameters to use for one audience. Finally, the method can (iv) help understand the effect of each parameter to allocate the budget to more important ones.

(i) **Real-time Ad Optimization.** For advertisers without strict budget constraints, the methodology can create the best ads for each user on a real-time basis. Taking into consideration the eco-system described in Chapter 1, once the DSP gets information from the publisher that an ad slot is available, the proposed methodology can create the ad best for that specific user.

(ii) **Find the best ad for a specific audience.** Using a specific set of users, the methodology is used to find the best parameters that fit a large proportion of the users. By doing so, the advertiser finds the best ad for that specific kind of users before informing the advertising network about the target audience.

(iii) **Find the best audience for a specific ad.** If the advertiser's resources are limited and he wants to optimise the reach of few parameters, the advertiser can set these parameters and optimise for different kind

of users. The results will show different predicted CTRs. The advertiser can then analyse the characteristics of the users that received a high CTR and use them as an audience for that specific ad.

(iv) **Parameters Budget Allocation.** Because of the capacity to incorporate constraints, advertisers can test for different constraints and understand the effect of each one. If subject to a budget limit, the advertisers can allocate the budget into developing parameters that are more likely to affect the CTR.

While (i) helps optimise the ads on a real-time basis, (ii), (iii), and (iv) provide tools to better understand the parameters of an ad as well as the target audience. In that sense, the methodology can be used as part of the whole marketing campaign decision making instead of being a real-time optimisation method only.

5.2 Implications for Research

The proposed methodology raises a number of potential research areas. Section 2.1 showed that the use of metaheuristics in a marketing optimisation context is limited. Thus, the methodology reveals the potential benefits of using a metaheuristic for the various research cited in Section 2.1.

Because of the use of a fitness function in GA, the proposed methodology's function can be altered to introduce various optimisation goals. Thus, the optimisation of conversion rates (CR) can be explored instead of the CTR. More sophisticated fitness functions able to compute various measures can also be included. In fact, another interesting research would be to include both the CTR and the CR in the same fitness function by using two different predictors; the first predictor would predict the CTR based on users that previously saw the ads while the second would predict the CR based on the users that bought products without necessarily seeing the ad. By doing so, it would be possible to make use of the characteristics of people who visit the website and the users who saw the ad independently and eventually improve the prediction power.

In the same way Miralles-Pechuán et al. (Miralles-Pechuán et al. 2018), Deane (Deane 2012) and this paper make use of a metaheuristic, the other facets of marketing campaign optimisation can also be explored using GA. Regarding the reach optimisation problem discussed in Section 2.1.3, GA can be used to find the best channels; each chromosome can constitute a different channel and the fitness function can compute the estimated number of users reached. This can potentially increase the scale of selection even further than the methodology proposed Danaher et al. (Danaher et al. 2010) and later improved by Paulson et al. (Paulson et al. 2015). The revenue optimisation discussed in Section 2.1.4 problem can be tackled by modifying the fitness function of the GA to take into consideration the expected revenue for each campaign, channels, or time allocation. More precisely, this methodology can be included in Ren et al.'s technique who make use of bidding to optimise the profits of a given campaign (Ren et al. 2018). Thus, while Ren et al.'s methodology computes the optimal bid price, this methodology can build the right ad for each user.

On a more general perspective, the use of other single solution and population-based metaheuristics can be explored to determine which type leads to better solutions for marketing campaigns optimisation. In this context, the use classic metaheuristics such as Simulated Annealing (SA) (Kirkpatrick et al. 1983), Particle Swarm Optimization (PSO) (Eberhart & Kennedy 1995), and Ant Colony Optimization (ACO) (Colorni et al. 1991) can be explored as well as hyperheuristics.

Chapter 6

Conclusion

This report aims to provide advertisers with a methodology that dynamically creates adequate ads for each user. Hence, the methodology presented in this paper combines an ANN and a GA. The ANN predicts the CTR for a specific user-ad matching while the GA updates the ads' parameters to improved the predicted CTR. Leading to an increase in the quality of ads displayed based on the viewer.

A gap in the literature was identified where the use of metaheuristics is limited in a marketing campaign optimisation context in Section 2.1. The adequate model for CTR prediction was identified as an ANN in Section 2.2 and tuned in Section 4.3 to result in a four-layer model achieving an AUC of 0.719. The model was then included in a GA with a mutation and crossover rates of 0.025 and 0.8 respectively. Testing the methodology on a real-life dataset has shown a significant improvement over randomly assigning the ads for users. The average CTR for a random assignment is 36% and was increased to 91.5% after the optimisation. Moreover, the methodology proved to be efficient when testing on new users that were not present in the training dataset as well as setting a 50% constraints on the ads' features.

Overall, the methodology gives the possibility to improve the success of marketing campaigns while improving the experience of users online. Compared to a one-fits-all strategy, the advertiser correctly makes use of each available slot in the publishers' website. Additionally, and because of its structure to deal with each user independently, the methodology was found to have other uses from a business point of view. The first use is to find the right audience for one advertisement by optimising for a large set of users and selecting those with the highest CTR. Finding the right ad by optimising with several parameters and selecting the prevalent ones. Lastly, iteratively fixing each parameter to understand their individual impact and assign higher budgets to the ones that have more impact on the CTR.

6.1 Limitations

The first limitation of the research is the lack of information provided regarding the dataset used. Because the dataset is public, the ads' parameters were transformed before the dataset was published online. Thus, when optimising, it is assumed the features are independent of one another. However, if the features were made public, some additional constraints might be added. However, experiment 3 showed that the methodology is able to handle various constraints. Thus, if constraints are added, the effects will be a slight reduction

in CTR but not a failure of the methodology. The anonymised features also affected possible experiments with entirely new values for ads parameters by understanding each feature and creating new possible values.

The second limitation is related to the hardware used during the research that is very limited. Because of that, a sample needed to be selected instead of using the complete 40 million rows of data and try more advanced techniques for feature engineering. The selection of a sample was related to the fact that not all models could be trained in an online fashion like ANN and required the complete dataset to be loaded into memory to be trained. The hardware also affected the running time per user that averaged 400 ms but can be faster if using a better setup.

6.2 Future Work

Understanding the impact of each feature on the predicted CTR in the ANN is the principal future work. Doing so will lead to a better understanding of what are the users' characteristics that work best with some ads' parameters. Thus, giving the possibility to advertisers to understand the effect of each parameter in detail and adapt to their audience.

The CTR predictor represents the core of the methodology. Because of this, future work can focus on improving it by using different models like Long-Short Term Memory (LSTM) that have not been explored by researchers. The use of ensembles can also be beneficial; where different models are used to predict at the same time and the results are merged to assign a final value. The ensemble can be a group of ANNs with different parameters or different models such as LR, SVR, and RF.

Additionally, the use of more advanced fitness functions might also be explored by either optimising for a different goal such as the CR or including different metrics in the function. Thus, allowing the advertisers to optimise for different targets at the same time and achieve the desired performance for each campaign.

Finally, the methodology can be extended to include bidding calculations. In fact, one of the methodologies described in Section 2.1 can be used to complete the next step of the methodology which is computing the right bid for the optimised ad. In that sense, it would be interesting to make use of a reinforcement learning because of the interest it has received during the last few years as well as its efficiency.

References

Abbassi, Z., Bhaskara, A. & Misra, V. (2015), Optimizing display advertising in online social networks, *in* 'Proceedings of the 24th International Conference on World Wide Web', International World Wide Web Conferences Steering Committee, pp. 1–11.

Agarwal, D., Chen, B.-C. & Elango, P. (2009), Spatio-temporal models for estimating click-through rate, *in* 'Proceedings of the 18th international conference on World wide web', ACM, pp. 21–30.

Agarwal, D., Ghosh, S., Wei, K. & You, S. (2014), Budget pacing for targeted online advertisements at linkedin, *in* 'Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 1613–1619.

Aksakallı, V. (2012), 'Optimizing direct response in internet display advertising', *Electronic Commerce Research and Applications* **11**(3), 229–240.

Amin, K., Kearns, M., Key, P. & Schwaighofer, A. (2012), 'Budget optimization for sponsored search: Censored learning in mdps', *arXiv preprint arXiv:1210.4847* .

Aryafar, K., Guillory, D. & Hong, L. (2017), An ensemble-based approach to click-through rate prediction for promoted listings at etsy, *in* 'Proceedings of the ADKDD'17', ACM, p. 10.

Avazu Click Through Dataset (2015), `https://www.kaggle.com/c/avazu-ctr-prediction/data`.

Bengio, Y., Delalleau, O. & Roux, N. L. (2006), The curse of highly variable functions for local kernel machines, *in* 'Advances in neural information processing systems', pp. 107–114.

Bengio, Y. et al. (2009), 'Learning deep architectures for ai', *Foundations and trends® in Machine Learning* **2**(1), 1–127.

Bharadwaj, V., Chen, P., Ma, W., Nagarajan, C., Tomlin, J., Vassilvitskii, S., Vee, E. & Yang, J. (2012), Shale: an efficient algorithm for allocation of guaranteed display advertising, *in* 'Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 1195–1203.

Branco, P., Ribeiro, R. P. & Torgo, L. (2016), 'Ubl: an r package for utility-based learning', *arXiv preprint arXiv:1604.08079* .

Cai, H., Ren, K., Zhang, W., Malialis, K., Wang, J., Yu, Y. & Guo, D. (2017), Real-time bidding by reinforcement learning in display advertising, *in* 'Proceedings of the Tenth ACM International Conference on Web Search and Data Mining', ACM, pp. 661–670.

Chapelle, O., Manavoglu, E. & Rosales, R. (2015), 'Simple and scalable response prediction for display advertising', *ACM Transactions on Intelligent Systems and Technology (TIST)* **5**(4), 61.

Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2002), 'Smote: synthetic minority over-sampling technique', *Journal of artificial intelligence research* **16**, 321–357.

Chen, J.-H., Zhao, Z.-Q., Shi, J.-Y. & Zhao, C. (2017), 'A new approach for mobile advertising click-through rate estimation based on deep belief nets', *Computational intelligence and neuroscience* **2017**.

Chen, J., Sun, B., Li, H., Lu, H. & Hua, X.-S. (2016), Deep ctr prediction in display advertising, *in* 'Proceedings of the 2016 ACM on Multimedia Conference', ACM, pp. 811–820.

Chen, Q., Guo, Z., Dong, W. & Jin, L. (2018), Ads' click-through rates predicting based on gated recurrent unit neural networks, *in* 'AIP Conference Proceedings', Vol. 1967, AIP Publishing, p. 040056.

Chen, Y., Pavlov, D. & Canny, J. F. (2009), Large-scale behavioral targeting, *in* 'Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 209–218.

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M. et al. (2016), Wide & deep learning for recommender systems, *in* 'Proceedings of the 1st Workshop on Deep Learning for Recommender Systems', ACM, pp. 7–10.

Chickering, D. M. & Heckerman, D. (2003), 'Targeted advertising on the web with inventory management', *Interfaces* **33**(5), 71–77.

Chollet, F. et al. (2015), 'Keras', `https://keras.io`.

Colorni, A., Dorigo, M., Maniezzo, V. et al. (1991), Distributed optimization by ant colonies, *in* 'Proceedings of the first European conference on artificial life', Vol. 142, Paris, France, pp. 134–142.

Covington, P., Adams, J. & Sargin, E. (2016), Deep neural networks for youtube recommendations, *in* 'Proceedings of the 10th ACM Conference on Recommender Systems', ACM, pp. 191–198.

Da Silva, I. N., Spatti, D. H., Flauzino, R. A., Liboni, L. H. B. & dos Reis Alves, S. F. (2017), 'Artificial neural networks', *Cham: Springer International Publishing* .

Danaher, P. J., Lee, J. & Kerbache, L. (2010), 'Optimal internet media selection', *Marketing Science* **29**(2), 336–347.

Deane, J. (2012), 'Hybrid genetic algorithm and augmented neural network application for solving the online advertisement scheduling problem with contextual targeting', *Expert Systems with Applications* **39**(5), 5168–5177.

Dhanani, J. & Rana, K. (2018), Logistic regression with stochastic gradient ascent to estimate click through rate, *in* 'Information and Communication Technology for Sustainable Development', Springer, pp. 319–326.

Eastlake 3rd, D. & Jones, P. (2001), Us secure hash algorithm 1 (sha1), Technical report.

Eberhart, R. & Kennedy, J. (1995), A new optimizer using particle swarm theory, *in* 'Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on', IEEE, pp. 39–43.

Effendi, M. J. & Ali, S. A. (2016), 'Click through rate prediction for contextual advertisment using linear regression', *CoRR* **abs/1701.08744**.

Evans, D. S. (2008), 'The economics of the online advertising industry', *Review of network economics* **7**(3).

Fang, Z., Yue, K., Zhang, J., Zhang, D. & Liu, W. (2014), 'Predicting click-through rates of new advertisements based on the bayesian network', *Mathematical problems in engineering* **2014**.

Gabrilovich, E., Broder, A., Fontoura, M., Joshi, A., Josifovski, V., Riedel, L. & Zhang, T. (2009), 'Classifying search queries using the web as a source of knowledge', *ACM Transactions on the Web (TWEB)* **3**(2), 5.

Gao, Z. & Gao, Q. (2013), 'Ad-centric model discovery for prediciting ads's click-through rate', *Procedia Computer Science* **19**, 155–162.

Goldfarb, A. (2014), 'What is different about online advertising?', *Review of Industrial Organization* **44**(2), 115–129.

Goodrich, M. T. & Tamassia, R. (2008), *Data structures and algorithms in Java*, John Wiley & Sons.

Graepel, T., Candela, J. Q., Borchert, T. & Herbrich, R. (2010), Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine, Omnipress.

Guo, H., Tang, R., Ye, Y., Li, Z. & He, X. (2017), 'Deepfm: A factorization-machine based neural network for ctr prediction', *arXiv preprint arXiv:1703.04247* .

He, X., Pan, J., Jin, O., Xu, T., Liu, B., Xu, T., Shi, Y., Atallah, A., Herbrich, R., Bowers, S. et al. (2014), Practical lessons from predicting clicks on ads at facebook, *in* 'Proceedings of the Eighth International Workshop on Data Mining for Online Advertising', ACM, pp. 1–9.

IAB (2017), 'Iab internet advertising revenue report conducted by pricewaterhousecoopers (pwc)'.

Jiang, Z. (2016), 'Research on ctr prediction for contextual advertising based on deep architecture model', *Journal of Control Engineering and Applied Informatics* **18**(1), 11–19.

Jiang, Z., Gao, S. & Li, M. (2018), 'An improved advertising ctr prediction approach based on the fuzzy deep neural network', *PloS one* **13**(5), e0190831.

Jin, J., Song, C., Li, H., Gai, K., Wang, J. & Zhang, W. (2018), 'Real-time bidding with multi-agent reinforcement learning in display advertising', *arXiv preprint arXiv:1802.09756* .

Juan, Y., Zhuang, Y., Chin, W.-S. & Lin, C.-J. (2016), Field-aware factorization machines for ctr prediction, *in* 'Proceedings of the 10th ACM Conference on Recommender Systems', ACM, pp. 43–50.

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983), 'Optimization by simulated annealing', *science* **220**(4598), 671–680.

Kondakindi, G., Rana, S., Rajkumar, A., Ponnekanti, S. K. & Parakh, V. (2014), 'A logistic regression approach to ad click prediction', *Mach. Learn.-Cl. Proj* .

König, A. C., Gamon, M. & Wu, Q. (2009), Click-through prediction for news queries, *in* 'Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval', ACM, pp. 347–354.

Korula, N., Mirrokni, V. S. & Nazerzadeh, H. (2016), 'Optimizing display advertising markets: Challenges and directions.', *IEEE Internet Computing* **20**(1), 28–35.

Kramer, O. (2017), *Genetic algorithm essentials*, Vol. 679, Springer.

Kriesel, D. (2007), 'A brief introduction on neural networks'.

Kumar, R., Naik, S. M., Naik, V. D., Shiralli, S., Sunil, V. & Husain, M. (2015), Predicting clicks: Ctr estimation of advertisements using logistic regression classifier, *in* '2015 IEEE International Advance Computing Conference (IACC)'.

Langheinrich, M., Nakamura, A., Abe, N., Kamba, T. & Koseki, Y. (1999), 'Unintrusive customization techniques for web advertising', *Computer Networks* **31**(11-16), 1259–1272.

Larochelle, H., Bengio, Y., Louradour, J. & Lamblin, P. (2009), 'Exploring strategies for training deep neural networks', *Journal of machine learning research* **10**(Jan), 1–40.

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

Lee, K.-C., Jalali, A. & Dasdan, A. (2013), Real time bid optimization with smooth budget delivery in online advertising, *in* 'Proceedings of the Seventh International Workshop on Data Mining for Online Advertising', ACM, p. 1.

Li, P., Shrivastava, A., Moore, J. L. & König, A. C. (2011), Hashing algorithms for large-scale learning, *in* 'Advances in neural information processing systems', pp. 2672–2680.

Lin, C.-C., Chuang, K.-T., Wu, W. C.-H. & Chen, M.-S. (2016), Combining powers of two predictors in optimizing real-time bidding strategy under constrained budget, *in* 'Proceedings of the 25th ACM International on Conference on Information and Knowledge Management', ACM, pp. 2143–2148.

Ling, C. X. & Li, C. (1998), Data mining for direct marketing: Problems and solutions., *in* 'Kdd', Vol. 98, pp. 73–79.

Maas, A. L., Hannun, A. Y. & Ng, A. Y. (2013), Rectifier nonlinearities improve neural network acoustic models, *in* 'Proc. icml', Vol. 30, p. 3.

McMahan, H. B., Holt, G., Sculley, D., Young, M., Ebner, D., Grady, J., Nie, L., Phillips, T., Davydov, E., Golovin, D. et al. (2013), Ad click prediction: a view from the trenches, *in* 'Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 1222–1230.

Miralles-Pechuán, L., Ponce, H. & Martínez-Villaseñor, L. (2018), 'A novel methodology for optimizing display advertising campaigns using genetic algorithms', *Electronic Commerce Research and Applications* **27**, 39–51.

Miralles-Pechuán, L., Rosso, D., Jiménez, F. & García, J. M. (2017), 'A methodology based on deep learning for advert value calculation in cpm, cpc and cpa networks', *Soft Computing* **21**(3), 651–665.

Mitchell, M. (1998), *An introduction to genetic algorithms*, MIT press.

Mookerjee, R., Kumar, S. & Mookerjee, V. S. (2012), 'To show or not show: Using user profiling to manage internet advertisement campaigns at chitika', *Interfaces* **42**(5), 449–464.

Muthukrishnan, S., Pál, M. & Svitkina, Z. (2007), Stochastic models for budget optimization in search-based advertising, *in* 'International Workshop on Web and Internet Economics', Springer, pp. 131–142.

Nair, V. & Hinton, G. E. (2010), Rectified linear units improve restricted boltzmann machines, *in* 'Proceedings of the 27th international conference on machine learning (ICML-10)', pp. 807–814.

Nakamura, A. & Abe, N. (2005), 'Improvements to the linear programming based scheduling of web advertisements', *Electronic Commerce Research* **5**(1), 75–98.

Pan, J., Xu, J., Ruiz, A. L., Zhao, W., Pan, S., Sun, Y. & Lu, Q. (2018), Field-weighted factorization machines for click-through rate prediction in display advertising, *in* 'Proceedings of the 2018 World Wide Web Conference on World Wide Web', International World Wide Web Conferences Steering Committee, pp. 1349–1357.

Paulson, C., Luo, L. & James, G. M. (2015), 'Optimal large-scale internet media selection', *under preparation for 2nd round review, Journal of Marketing Research.* ASA Statistics in Marketing Travel Award* .

Pepelyshev, A., Staroselskiy, Y. & Zhigljavsky, A. (2015), Adaptive targeting for online advertisement, *in* 'International Workshop on Machine Learning, Optimization and Big Data', Springer, pp. 240–251.

Perlich, C., Dalessandro, B., Hook, R., Stitelman, O., Raeder, T. & Provost, F. (2012), Bid optimizing and inventory scoring in targeted online advertising, *in* 'Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 804–812.

Provost, F. (2000), Machine learning from imbalanced data sets 101, *in* 'Proceedings of the AAAI'2000 workshop on imbalanced data sets', pp. 1–3.

Provost, F. & Fawcett, T. (2001), 'Robust classification for imprecise environments', *Machine learning* **42**(3), 203–231.

Qu, Y., Cai, H., Ren, K., Zhang, W., Yu, Y., Wen, Y. & Wang, J. (2016), Product-based neural networks for user response prediction, *in* 'Data Mining (ICDM), 2016 IEEE 16th International Conference on', IEEE, pp. 1149–1154.

Ren, K., Zhang, W., Chang, K., Rong, Y., Yu, Y. & Wang, J. (2018), 'Bidding machine: Learning to bid for directly optimizing profits in display advertising', *IEEE Transactions on Knowledge and Data Engineering* **30**(4), 645–659.

Ren, K., Zhang, W., Rong, Y., Zhang, H., Yu, Y. & Wang, J. (2016), User response learning for directly optimizing campaign performance in display advertising, *in* 'Proceedings of the 25th ACM International on Conference on Information and Knowledge Management', ACM, pp. 679–688.

Rendle, S. (2010), Factorization machines, *in* 'Data Mining (ICDM), 2010 IEEE 10th International Conference on', IEEE, pp. 995–1000.

Richardson, M., Dominowska, E. & Ragno, R. (2007), Predicting clicks: estimating the click-through rate for new ads, *in* 'Proceedings of the 16th international conference on World Wide Web', ACM, pp. 521–530.

Sasaki, Y. et al. (2007), 'The truth of the f-measure', *Teach Tutor mater* **1**(5), 1–5.

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

Ta, A.-P. (2015), Factorization machines with follow-the-regularized-leader for ctr prediction in display advertising, *in* 'Big Data (Big Data), 2015 IEEE International Conference on', IEEE, pp. 2889–2891.

Trofimov, I., Kornetova, A. & Topinskiy, V. (2012), Using boosted trees for click-through rate prediction for sponsored search, *in* 'Proceedings of the Sixth International Workshop on Data Mining for Online Advertising and Internet Economy', ACM, p. 2.

Walczak, S. (2018), Artificial neural networks, *in* 'Encyclopedia of Information Science and Technology, Fourth Edition', IGI Global, pp. 120–131.

Wang, F., Suphamitmongkol, W. & Wang, B. (2013), 'Advertisement click-through rate prediction using multiple criteria linear programming regression model', *Procedia Computer Science* **17**, 803–811.

Weinberger, K., Dasgupta, A., Langford, J., Smola, A. & Attenberg, J. (2009), Feature hashing for large scale multitask learning, *in* 'Proceedings of the 26th annual international conference on machine learning', ACM, pp. 1113–1120.

Whitley, D. (1994), 'A genetic algorithm tutorial', *Statistics and computing* **4**(2), 65–85.

Wu, D., Chen, X., Yang, X., Wang, H., Tan, Q., Zhang, X. & Gai, K. (2018), 'Budget constrained bidding by model-free reinforcement learning in display advertising', *arXiv preprint arXiv:1802.08365* .

Xu, J., Lee, K.-c., Li, W., Qi, H. & Lu, Q. (2015), Smart pacing for effective online ad campaign optimization, *in* 'Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', ACM, pp. 2217–2226.

Yan, J., Liu, N., Wang, G., Zhang, W., Jiang, Y. & Chen, Z. (2009), How much can behavioral targeting help online advertising?, *in* 'Proceedings of the 18th international conference on World wide web', ACM, pp. 261–270.

Yin, D., Mei, S., Cao, B., Sun, J.-T. & Davison, B. D. (2014), Exploiting contextual factors for click modeling in sponsored search, *in* 'Proceedings of the 7th ACM international conference on Web search and data mining', ACM, pp. 113–122.

Zhang, W., Yuan, S. & Wang, J. (2014), Optimal real-time bidding for display advertising, *in* 'Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 1077–1086.

Zhang, W., Zhang, Y., Gao, B., Yu, Y., Yuan, X. & Liu, T.-Y. (2012), Joint optimization of bid and budget allocation in sponsored search, *in* 'Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 1177–1185.

Zhang, Y., Dai, H., Xu, C., Feng, J., Wang, T., Bian, J., Wang, B. & Liu, T.-Y. (2014), Sequential click prediction for sponsored search with recurrent neural networks., *in* 'AAAI', Vol. 14, pp. 1369–1375.

Zhou, G., Song, C., Zhu, X., Ma, X., Yan, Y., Dai, X., Zhu, H., Jin, J., Li, H. & Gai, K. (2017), 'Deep interest network for click-through rate prediction', *arXiv preprint arXiv:1706.06978* .

Appendix A

Marketing Campaign Optimisation Papers

This appendix summarises the papers cited in Section 2.1 in terms of year, goal, and the use of ML methodologies.

Author	**Year**	**Goal**	**Machine Learning**
Langheinrich et al.	1999	Click-Through Rate Optimisation	No
Chickering & Heckerman	2003	Click-Through Rate Optimisation	No
Nakamura & Abe	2005	Click-Through Rate Optimisation	No
Muthukrishnan et al.	2007	Budget Optimisation	No
Yan et al.	2009	Click-Through Rate Optimisation	No
Chen et al.	2009	Click-Through Rate Optimisation	Yes
Danaher et al.	2010	Ad Allocation	No
Perlich et al.	2012	Bid Optimisation	Yes
Aksakallı	2012	Revenue Optimisation	No
Zhang et al.	2012	Revenue Optimisation	No
Mookerjee et al.	2012	Revenue Optimisation	Yes
Bharadwaj et al.	2012	Ad Allocation	No
Deane	2012	Budget Optimisation	Yes
Amin et al.	2012	Budget Optimisation	Yes
Lee et al.	2013	Budget Optimisation	No
Zhang, Yuan & Wang	2014	Bid Optimisation	Yes
Agarwal et al.	2014	Budget Optimisation	No
Paulson et al.	2015	Ad Allocation	No
Pepelyshev et al.	2015	Click-Through Rate Optimisation	Yes
Xu et al.	2015	Budget Allocation	Yes
Abbassi et al.	2015	Click-Through Rate Optimisation	No
Lin et al.	2016	Bid Optimisation	Yes
Ren et al.	2016	Revenue Optimisation	Yes
Cai et al.	2017	Bid Optimisation	Yes
Wu et al.	2018	Bid Optimisation	Yes
Jin et al.	2018	Bid Optimisation	Yes
Ren et al.	2018	Bid Optimisation	Yes
Miralles-Pechuán et al.	2018	Click-Through Rate Optimisation	Yes

Table A.1: Marketing Campaigns Optimisation Literature

Appendix B

CTR Prediction Papers

This appendix summarises the papers cited in Section 2.2 in terms of year and methodology used.

Author	**Year**	**Methodology**
Richardson et al.	2007	Logistic Regression
Agarwal et al.	2009	Dynamic Linear Regression
König et al.	2009	Multi-Additive Regression Tree
Graepel et al.	2010	Bayesian Networks
Trofimov et al.	2012	Boosted Trees
Gao & Gao	2013	Multi-variate Linear Regression
Wang et al.	2013	Multi-criteria Linear Regression
Kondakindi et al.	2014	Logistic Regression
He et al.	2014	Regression Trees and Linear Regression
Fang et al.	2014	Bayesian Networks
Yin et al.	2014	Multi-task Linear Regression
Kumar et al.	2015	Logistic Regression
Ta	2015	Factorization Machines
Chapelle et al.	2015	Logistic Regression
Juan et al.	2016	Factorization Machines
Chen et al.	2016	Deep Neural Networks
Effendi & Ali	2016	Linear Regression
Covington et al.	2016	Deep Neural Networks
Cheng et al.	2016	Wide&Deep Neural Networks
Jiang	2016	Deep Neural Networks & Logistic Regression
Qu et al.	2016	Deep Neural Networks
Guo et al.	2017	Deep Neural Networks & Factorization Machines
Miralles-Pechuán et al.	2017	Deep Neural Network
Zhou et al.	2017	Deep Neural Networks
Dhanani & Rana	2018	Logistic Regression
Pan et al.	2018	Factorization Machines
Jiang et al.	2018	Deep Neural Networks & Fuzzy Theory
Chen et al.	2018	Recurrent Neural Networks

Table B.1: Click-Through Rate Predictions Literature