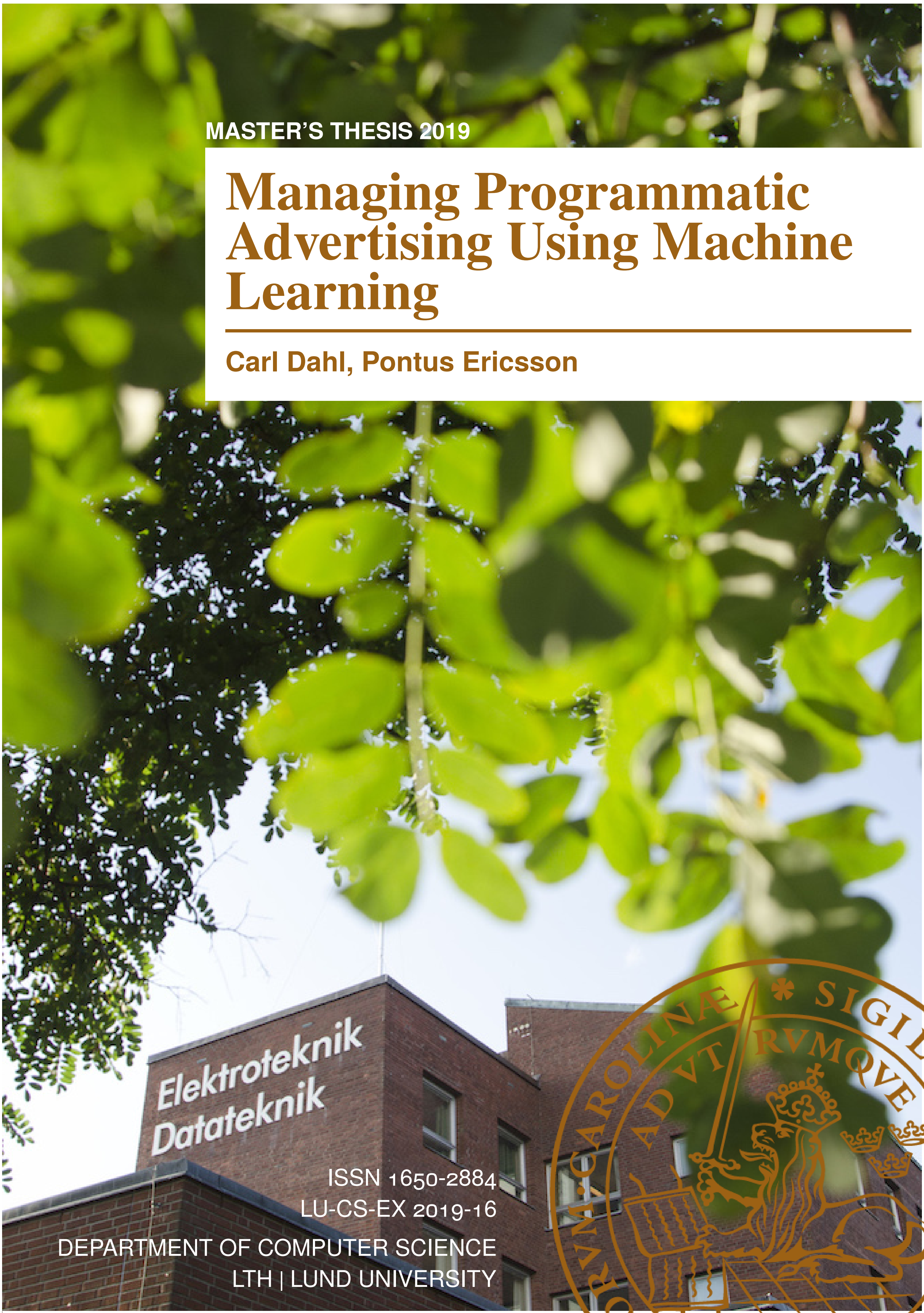


MASTER'S THESIS 2019

Managing Programmatic Advertising Using Machine Learning

Carl Dahl, Pontus Ericsson

ISSN 1650-2884
LU-CS-EX 2019-16
DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2019-16

Managing Programmatic Advertising Using Machine Learning

Carl Dahl, Pontus Ericsson

Managing Programmatic Advertising Using Machine Learning

Carl Dahl
dat14cda@student.lu.se

Pontus Ericsson
dat14pe1@student.lu.se

July 5, 2019

Master's thesis work carried out at Emerse Sverige AB.

Supervisors: Pierre Nugues, Pierre.Nugues@cs.lth.se
Carl-Johan Grund, carl-johan.grund@emerse.com
Rasmus Larsson, rasmus.larsson@emerse.com

Examiner: Jacek Malec, Jacek.Malec@cs.lth.se

Abstract

This thesis is an exploratory study into the possibility of using machine learning to manage advertisement campaigns and agents involved in real-time bidding. The norm for the industry of real time bidding is currently having human operators managing campaigns by changing settings to maximize the number of clicks. The goal was to investigate the possibility of automating this process, to at the very least assist the human operators with making better decisions. The first part of the project was to build a model for predicting the click-through rate (CTR) of the ad campaigns. The second part was to use the model to suggests optimal settings for bidding agents. The outcome was a model with an accuracy of 92% in predicting whether an ad was to generate any clicks or not, and with an accuracy of 58% to predict the outcome of an agent in the different categories "few clicks", "some clicks" and "many clicks". We tried the model with eight live tests giving varying results, but overall the average CTR of the test agents was 8% better when compared to a test agent created by a human operator. These tests show that the model carries potential, but needs further development to become more stable. Unexpectedly using the images in the ad banners as part of the input to train models resulted in a lower accuracy compared to without. This was however likely due to the small amount of training data. The conclusion for the project is that the method of automating advertisement could be utilized with much success but requires large datasets.

Keywords: Machine Learning, Classification, Programmatic Advertising

Acknowledgements

This master thesis was done at Emerse Sverige AB and would not have been possible without their help and access to their platform and data. We want to thank Carl-Johan Grund and Rasmus Larsson for the opportunity and the help along the way. We also want to thank the people working in ad-ops at Emerse for helping us with the live tests during our project.

A final thanks our supervisor Pierre Nugues for his support throughout the entire project and for taking the time for weekly meetings in the beginning of the project to make sure we got started on the right track. He gave us many tips on how to filter and prepare our data, and how to improve our models.

Contents

Chapter 1
Introduction

Online advertising is a fast growing business. In the first six months of 2018 in the United States, internet advertising revenues totaled 49.5 billion dollars. Roughly a 23% increase from the year before (Silverman and Gaffney, 2018). The latest statistics for the market in Europe comes from 2017 where the revenue totaled 12 billion euro, an increase from 2016 by 27% (Puffett, 2018).

This project was done at Emerse Sverige AB who develops its own *demand side platform (DSP)*, which is an online tool for companies to buy ad-spaces. Emerse also develops algorithms for the automated purchase of online *ad spaces* through programmatic exchanges. A major part of the company is managing different advertisement *campaigns*. This is done through the DSP by adjusting different parameters such as the pacing of the budget, what age group to target, and which websites to advertise on, to name a few.

This master thesis explores the possibility of using a machine learning model to predict the optimal settings for such parameters given an ad campaign. The goal for the model is to perform at least as good, and hopefully better than a human operator. The purpose of the model is not to replace human operators, but rather to automate some parts of the daily work of adjusting different campaigns. This would reduce the effort required for each campaign, which in turn allows one operator to manage more campaigns at the same time.

1.1 Vocabulary

This section will give the definitions of the words and abbreviations used throughout the report. The first time a word in the vocabulary is mentioned in the report it is shown in *italics* as an indication.

Word	**Definition**
Ad space	A place on a website or app reserved for an ad. Theses spaces are what are sold on programmatic exchanges. Each ad space has a predetermined size.
Agent	A managing instance of an ad that has assigned values such as daily budget and target audience which allows for deciding what ad spaces to bid on online.
Assurance value	A percentage of certainty used by the program to establish the best variable settings. Explained further in Section 2.6.1
Campaign	A set of creatives and agents with the same goal for the same client can be described as a campaign. One campaign could be to advertise a new version of a product, for which multiple different ad banners and videos could be used.
Categorical data	Data from the agents features that can be categorized i.e non numerical values, such as budget, IAB category, pacing type etc.
Click-through rate (CTR)	Defined as the number of unique clicks divided by the number of impressions. This metric shows the percentage of people who saw the ad and clicked on it.
Constant	A feature of an agent that can not be changed when optimizing. Examples of this is budget and IAB category
Convolutional neural network (CNN)	A neural network with convolutional layers to extract features from images.
Creative	An image or video of the ad. If an ad banner has multiple different sizes then each one is its own creative. If an ad has both an image and a video then they are different creatives.
Demand side platform (DSP)	A tool used by companies looking to buy ad spaces. DSPs are connected to multiple programmatic exchanges and allows for RTB.
Interactive advertising bureau (IAB)	An recognized international group of marketing companies which sets standards and best practices of the industry.
Impression	A measurement of how many times an ad is shown to a user.
Model	A machine learning model that predicts agent performance. *The model* refers to the one designed in this thesis if not otherwise described.
Mutant	An instance of the agent the program is currently optimizing, with some variables changed.

Table 1.1: Vocabulary and abbreviations table

Word	**Definition**
Program	The entire end-product of the thesis, being a system that utilizes the model to optimize agent performance rather than just predict it.
Real-Time Bidding (RTB)	Bidding on programmatic exchanges in real time.
Supply Side Platform (SSP)	A tool used by companies looking to sell ad spaces. Like DSPs, they are also connected to multiple programmatic exchanges.
Variable	A model feature from an agent which can be changed to optimize the performance. Examples of this is Pacing type and Optimization strategy.
Unique Clicks	Number of unique people that have clicked on an ad. The number of times an ad has been clicked in total is just referred to as *clicks*.

Table 1.2: Vocabulary and abbreviations continued

1.2 Problem statement

The objective of the thesis is to explore how machine learning can be used to increase effectiveness for the online marketing business. In practice, we did this by designing a *program* that assists with optimizing advertisement performance at an online marketing company. As such, a more practical statement of the problem is divided into two steps:

1. The first step is to design a machine learning *model* that is trained to predict the final *click-through rate (CTR)* for a certain *agent* based on both the settings and situational *constants* of the agent.

2. The second step is to create an algorithm that takes an agent with all its relevant features, and for each of the different *variables*, such as active hours and device preference, predicts the CTR using the machine learning model and determines the optimal settings of the agent.

We used the CTR to measure performance, but other metrics exist. For example, the number of *impressions* or cost per click for the agent. We however determined that these were too closely associated with the budget. We hypothesized that CTR was more affected by changes to the other settings of the agent and therefore more relevant to the model.

1.3 Online advertising

As previously stated online advertising, or internet advertising, is a multi-billion dollar industry and in this section we will briefly describe how programmatic advertising works.

In 2017, about 62% of European online ads were traded programmatically (Puffett, 2018). This refers to what is also known as *real-time bidding (RTB)*, where ad spaces are bought on programmatic exchanges (Yuan et al., 2013).

RTB is exactly what the name suggests: Bidding in real time on ad spaces. It emerged in 2009 according to Yuan et al. (2013). When a user opens, for example, a website with an ad space, a bid request for that impression will be created and sent to an ad exchange. These exchanges show all current bid requests along with some meta data regarding the user. All advertisers connected to that ad exchange can see the bid request and can then choose to place a bid. The ad exchange will then check all the bids received on the request and decide the winner. The winning advertiser will send their ad so that it can be shown to the user (Yuan et al., 2013).

These ad exchanges, as mentioned by Yuan et al. (2013), consist of multiple different ad network publishers, who offer to sell impressions. Ad exchanges are a way to combine ad networks to make it easier for advertiser to buy fitting impressions. They also mitigate the inconvenience of registering to every ad network, and instead allow for simply registering to the ad exchange.

In order for advertisers to take advantage of RTB on ad exchanges, they work with a third party platform referred to as a demand side platform, or DSP. The DSP will act as a delegate to the advertiser and will bid on bid request interesting to the advertiser. Figure 1.1 illustrates how the different parties in online advertising are connected.

Yuan et al. (2013) lists three advantages of using a DSP as compared to interacting with ad networks yourself:

1. Advertisers only have to register with the DSP and avoid having to manage many registrations with different ad networks.

2. DSPs allows for more detailed optimization and information, thanks to local logs instead of periodic reports from ad networks.

3. DSPs allows for more customization in terms of the campaign. One example being that DSPs can track the unique id of users to limit the amount of impressions for each user. As such it can restrict the showing of one specific ad to x times per day/week/month to the same person. Another example is to show ads based on the weather in the region of the user. This can be done thanks to the meta data from the bid request including both the coordinates and IP address of the user. The DSP can then simply look at whichever weather forecast site they prefer to determine if the user belongs to the target group of the ad.

On the opposite side, *supply side platforms (SSPs)* are a help to publishers by acting as central management consoles for different ad exchanges. Just like DSPs allow advertisers to customize their campaigns more, SSPs allow publishers to customize the impressions they sell. For example setting a preference of bidders (bid bias) (Yuan et al., 2013).

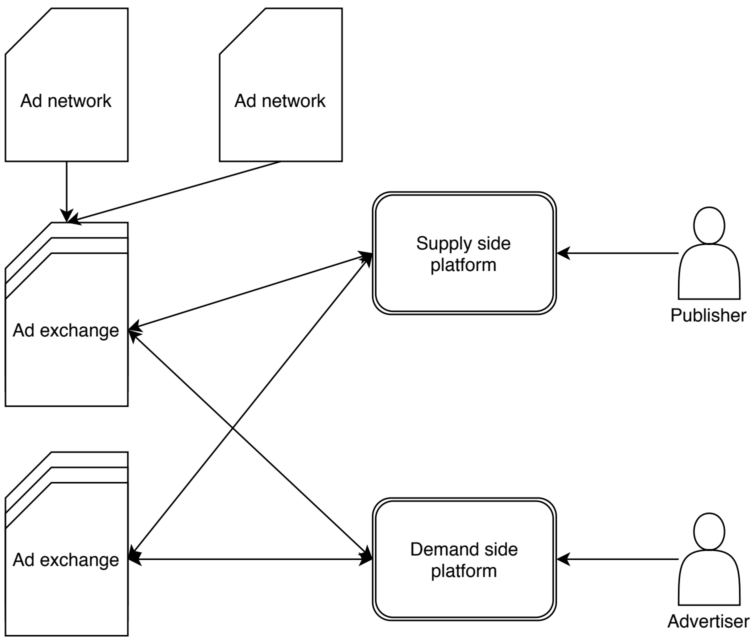


Figure 1.1: Overview of online advertising and the connections between different parties

1.4 The Interactive Advertising Bureau

As most online ad *creatives* are in the form of images and videos that need to fit a certain space on a website allocated for advertisements, it is important that the sizes match up. The *Interactive Advertising Bureau (IAB)* serves as an international source for guidelines on what ads should look like, what counts as an impression and view, and standardize sizes for ads and ad spaces (Interactive Advertising Bureau, 2019).

IAB have also designed a set of categories to divide ads into what type of product or service they are advertising. This list includes 26 different topics that includes but is not limited to: *Education*, *Health & Fitness* and *Society* (AerServ, 2018).

Whether the displaying of an ad counts as an impression or not is decided through a visibility ratio decided by IAB. This definition also depends on whether the ad is large or not.

A small ad is defined as an ad with a creative consisting of fewer than 242,500 pixels. This is equivalent to a 970x250 image. The ad must be in a visible space in an active window with at least 50% of the pixels showing for at least one continuous second. For a large ad of more than 242,500 pixels, at least 30% of the ad has to be visible for the full continuous second for it to count as a valid impression (MRC, 2014).

1.5 Previous work

There is a multitude of articles written on the subject of predicting CTR in online advertising. However, most of them tend to focus on predicting the CTR for a singular viewing of an ad, rather than the performance of an autonomous agent deciding which ad spaces to purchase (Chen et al., 2016; Graepel et al., 2010).

When it comes to using machine learning to improve online advertisement in more general terms, there are many scientific articles on the subject. Two examples of this are the papers “Ad analysis using machine learning by classifying and recommending advertisements for a given category of videos” by Kaushik et al. (2017) and “Real Life Machine Learning Case on Mobile Advertisement: A Set of Real-Life Machine Learning Problems and Solutions for Mobile Advertisement” by Seker (2016). The first paper, is an experimental analysis whether machine learning can be applied to make advertisements for videos better directed to the target audience. The second paper, is an output of data for a science study on a set of real life problems from the mobile advertisement industry.

Both of these papers conclude that there are a multitude of machine learning based solutions for these fields. They also argue for that these have good potential and that the results gained can benefit the advertisement industry at large.

Chapter 2
Method

In this chapter we present an analysis of the data used as well as what tools were used in the project. We also describe how we created the models and how we used them in order to optimize settings for agents.

2.1 The architecture of the market platform

An essential part to understand how to optimize ad performance is to look at what can be optimized and how it connects to the remaining system. For this problem, the focus is agents. The decisions on which impressions to buy are determined by the agents based on their settings. Each agent markets a certain campaign and is connected to one or more *creatives*, such as banners or videos. The more creatives, the more options for the agent on which impression to buy since the creative and the ad space size must match. A campaign can have multiple agents and in many cases, this is preferred. The benefit of creating multiple agents for the same ad is that then every agent can be customized to better target different kinds of users or websites. How an ad is split into multiple agents is shown in Figure 2.1.

Another advantage of this is that not all eggs are placed in the same basket so to speak; the risk of failure is spread out across multiple agents instead of focused on one. Since there is no magic set of settings that will always work, managing ad campaigns consists of making educated guesses based on experience in the field.

2.1.1 Agent features

Agents are the data points used as input in training the machine learning model. The agents contain many different parameters and a lot of them were ignored in training for different reasons. Only the parameters we actually used in the project are presented here, separated between *variables* and *constants*.

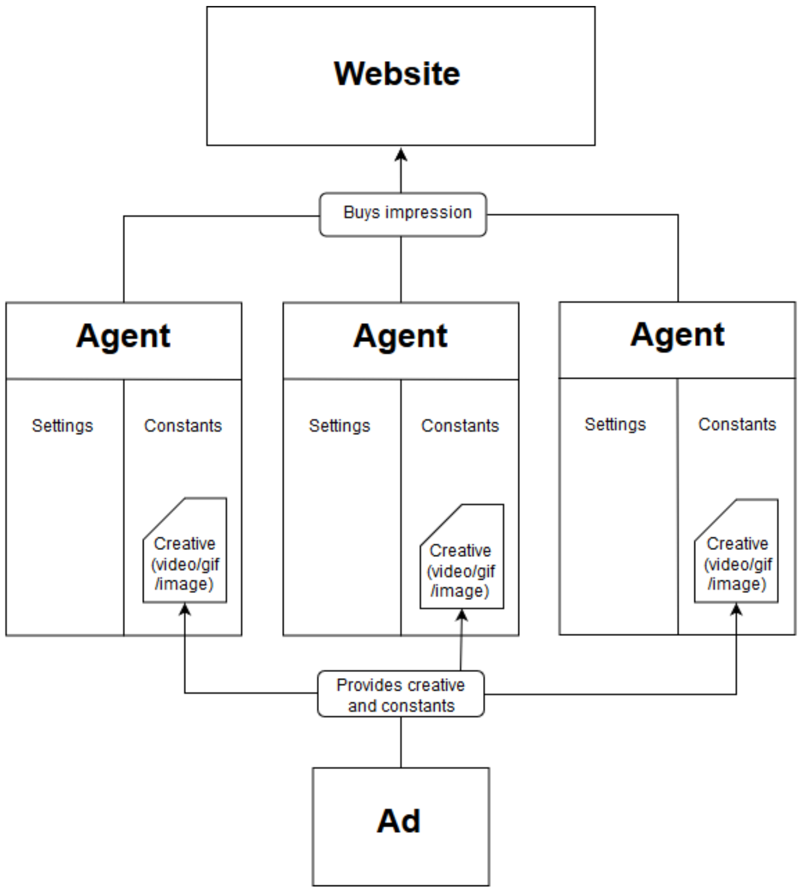

Website
Buys impression
Agent
Settings
Constants
Creative (video/gif /image)
Agent
Settings
Constants
Creative (video/gif /image)
Agent
Settings
Constants
Creative (video/gif /image)
Provides creative and constants
Ad

Figure 2.1: *A visualization of the agents relation to the ads and websites*

Constants

- **CTR:** As explained in the vocabulary, this is the clicks per impression ratio and is used as the target value for our model to optimize over.
- **Budget:** The amount of money an agent has available for bidding on impressions.
- **Start and end date/Flight time:** The dates from which the agent goes live and ends. In the model this is simplified as the number of days the agent was active, referred to as flight time.
- **IAB category:** The IAB category that the ad belongs to.
- **Is Video:** Whether the creative is an image/GIF or a playable video.
- **Width:** The width of the creative banner in pixels. If the creative is a video, width is always zero.
- **Height:** The height of the creative banner in pixels. If the creative is a video, height is always zero.
- **H/W ratio:** Width and height divided. If the creative is a video, H/W ratio is always zero.

Variables

- **Optimization Strategy:** A Boolean that dictates whether the bid on impressions should be a fixed sum of money for each impression, or if an algorithm should be used to determine the value of the impression so that a bid can be made thereafter.
- **Pacing type:** Can have one of three values, and dictates how the budget should be spent over the course of the campaign. The value can be either *"Even"* to spend the money evenly across the campaign, *"Fast"* to spend as much as possible in the beginning of the campaign, and *"Custom"* where a set daily budget is spent every day.
- **Hour-of-week filter:** A binary string of 168 bits, where each bit represents an hour of the week and whether the ad should be active during this hour. This setting was simplified to a binary string of 8 bits and how is mentioned later in this chapter.
- **Device type:** A binary string of seven bits where each bit represents an allowed type of device to buy impressions on. This includes phones, personal computers, tablets and so on.

2.2 Tools used

Tools we used that aided the project are mentioned here.

Scikit learn. Scikit learn is a package for python to implement and manage machine learning models. We used this to design both the initial prototype and the complete model. We also used Scikit learn in the preparation of data before training: For example scaling numerical data and one hot encoding *categorical data* (Pedregosa et al., 2011).

Keras. Keras is an API designed for making neural networks written in Python. We used Keras to create various image classification models and ensemble methods (Chollet et al., 2015).

ImageNet and ResNet-50. ImageNet is a large public database of images for training different machine learning models (Deng et al., 2009). There are millions of images in the set, all labeled with what they depict. At the time of writing this report, ImageNet currently contains 14,197,122 images (Fei-Fei et al., 2019).

In this project, we used a pre-trained network called ResNet-50 which is trained on ImageNet (He et al., 2015). It gets its name from the fact that it is a residual network that is 50 layers deep. A residual network is built up of blocks of layers where the idea is to add the input to the block again after passing through the layers in the block, also known as shortcut connections. This is shown in Figure 2.2. Thanks to the shortcut connections, residual networks allows for building deeper networks without a degradation problem, as stated by He et al. (2015).

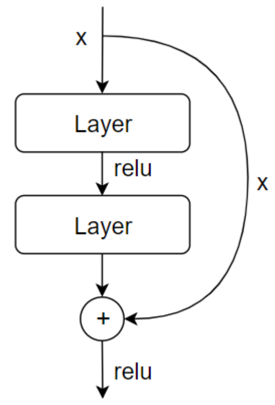


Figure 2.2: Simplified version of a residual building block

2.3 Analyzing the data

We extracted the data necessary for constructing the model from the database of the Emerse DSP. This consisted of 13,048 Agents each with their associated settings and results. In reality there were many more agents available that we decided not to use. The reasoning behind this comes from the advertisement market requiring new strategies constantly to stay productive (Louth, 1966). We suspected that using data from agents not intended to finish over the coming year would only serve to confuse the model and lessen the result. Agents older than 2 years were also removed for the same reason.

Agents with fewer impressions than 500 were also singled out as these were likely small-scale tests or agents terminated abruptly due to errors or mishappenings. There was also a handful of agents with null-values for some of the settings used, and were as such also removed.

2.3.1 Click-through rate

To get a proper understanding of the data, we plotted it to evaluate the distribution of CTR for all agents. The plot immediately uncovered two data points that were determined to be outliers, two agents with a CTR of 88% and 50% respectively. As all other agents have a CTR below 11% and a majority of them below 2%, the two outliers were removed. This plot with the outliers removed is displayed in Figure 2.3.

We later decided to only include agents with a CTR below 5% after speaking with experienced employees at Emerse. They explained that an unusually high CTR probably means that the agent is incorrect in some way. For example the ad could be misleading or in other ways tricking the user to click it.

Another noteworthy item regarding the data is the fact that over half the agents had a CTR of 0%. A result of having either zero clicks, or too few clicks compared to the agents impressions making the quotient almost zero. This was the reason why we later decided to divide the program into two models. One model classifying if the agent has a CTR of zero or not and the second one how good the CTR is.

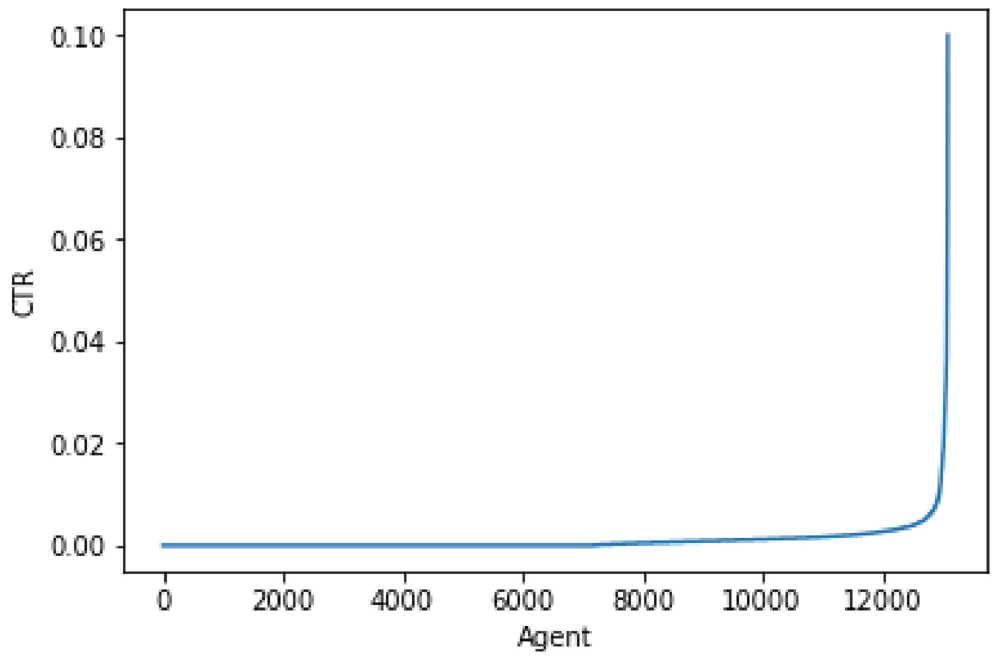


Figure 2.3: *A graph over the distribution of CTR for the agents.*

2.3.2 Feature selection

As the data contains multiple features, we had to use a selection process to determine the most useful ones. This process consisted mostly of experimentation, i.e. training the model with a new feature and see if the result is better or worse. We also spoke to the employees at Emerse to better understand which features were interesting.

At first we selected features mostly at random to create a base for the model. This naturally resulted in a low-accuracy model. Thus the features had to be looked at more closely and some needed to be adapted while others had to be removed entirely. An example of this is start and end dates of an agent that we adapted to a duration instead. The number of days an agent is active is more relevant than just a date, and it's easier to interpret for a machine learning model.

An issue common in machine learning, is when data points are missing some column of data. The problem is then how you represent this missing value, since all samples given to the model must have the same number of features.

An easy solution to this is to apply imputation. What this means is to replace all missing values with the average, or the most common value, depending on which is the most applicable to the feature. However, these solutions are counterproductive if a large part of the samples have inserted values. The data can become biased towards the most common value for example. There were a lot of different interesting settings available to agents, but since only a few of them used these settings it was impossible for a model to learn them effectively.

2.3.3 Hourly distribution

To optimize over what hours are effective to run an ad, it became essential to look more closely at the `hour-of-week filter` attribute. As this attribute was an array of 168 binary numbers, one-hot encoding it would result in the feature list being about 75% one hot-encoded hours. There was also a strong suspicion, that hours close to each other likely hosts similar audiences. Therefore, we plotted the data to look for trends, displayed in Figure 2.4.

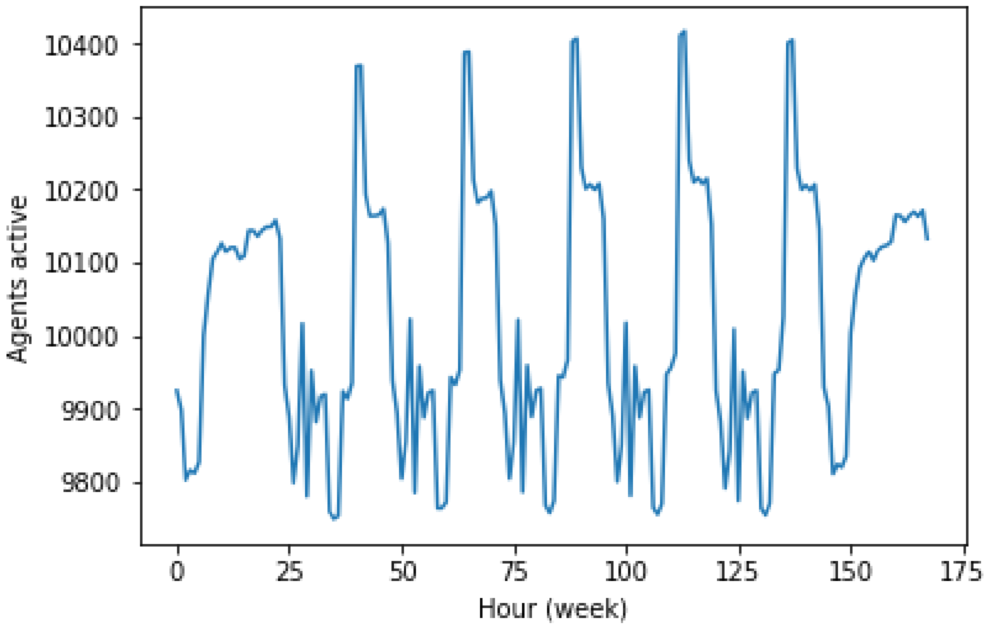


Figure 2.4: *A graph over the distribution of hours when an agent is active in order of Sunday to Saturday*

The plot shows a clear correlation of the hourly distribution between weekdays and weekends, and as such we decided to group all weekdays together and likewise group Saturday and Sunday together. We then simplified the data further by dividing each day into 4 categories. These are: Morning (07-11), Afternoon (12-16), Evening (17-22) and Night (23-06). This meant that the active times of one agent could be represented with 8 bits instead of 168, 4 for weekdays and 4 for weekends. If the agent had at least one active hour in a category, then that category will be marked as active, and all hours within that time frame will be treated as active.

2.4 Neural Networks

In the field of machine learning, one subfield is deep learning. The word “deep” in this case refers to the learning process being built up of layers with increasing importance. Deep learning is in practice almost always implemented with models called neural networks (Francois, 2017).

In this project we tried different versions neural networks with varying success. Here we will describe some key concepts in neural networks, starting with layers.

The layers in a neural network are the building blocks. There are different kinds but the general idea for one is to take an input, transform it in some way, and then return an output. The input is often a tensor, i.e. a matrix or vector. The output of a layer depends on its purpose. If its an intermediate layer in the model then the output will generally be a tensor. However, if its the final layer of the model then the output could be a real number in the case of regression, or a natural number representing a class in the case of classification. The transformation in a layer depends on the type of layer. In a Dense layer the transformation looks like this: $output = relu(dot(W, input) + b)$ (Francois, 2017). Breaking this down, the first thing that happens is a dot product between the input tensor and the tensor W. After that its an addition with a vector b, followed by a $relu$ operation which means to replace all negative numbers with zero. The W and b are attributes of the layer and are called the weights.

Before we define what the weights of a layer refers to, first we will describe the loss function. This function is the measurement of success during training in machine learning (Francois, 2017). After the model has turned input data into predictions, the loss function compares these predictions to the actual values and produces a loss value. In this project we used two different loss functions, categorical crossentropy for classification and mean-squared error for regression.

The next thing to understand is the weights in a layer. These are the learning part of machine learning. When training a model its the weights that change. When training a new model the weights are initially randomized. Obviously the initial values of the weight matrices are not very useful. They are in fact useless in predicting the desired data. But they are a starting point for the training. In training the weights are adjusted gradually over multiple iterations to find the optimal permutation. In order to know whether the values should be increased or decreased the gradient of the loss is calculated with regards to the weight matrix (Francois, 2017). The gradient is a generalisation of the derivative for multiple variables. This algorithm is called gradient descent.

In practice there are multiple ways to implement the gradient descent algorithm, in neural networks they are referred to as the optimizer of the model. In this project we mainly used RMSprop.

2.5 Building the model

The process of constructing the final model consisted of a lot of trial and error. In this section, we present this process and the decisions we made to arrive at the architecture of the model used in the end.

2.5.1 Regression

Initially we tried training multiple models, including linear regression, decision trees, and random forests. The resulting models performed rather poorly. We believe this was mainly due to the distribution of the dataset where over half the samples had a CTR of zero.

After multiple attempts, we finally concluded that regression was impractical in this project because of the low accuracy. We therefore decided to exclusively use classification.

The final version of the regression model we used was a neural network built in Keras. The model consisted of 3 dense layers, followed by a dropout of 5% and then another dense layer as output. Multiple other setups were attempted including more dense layers and a larger dropout but this is the one that resulted in the highest accuracy. The result of this can be seen in Section 3.1.1.

2.5.2 Classification

As can be seen in Figure 2.3 where the CTR is plotted, there is a large portion of the data samples that have a CTR of zero. The result of this is that the model had trouble to estimate the exact values as half the time the value was just zero. To mitigate this we trained a model to classify the data into two categories, zero or not. This model was used to filter out samples predicted to have a CTR of zero, so we could add a second model which we only trained on non-zero CTR samples. The connection between these two models is shown in Figure 2.5.

The second model, trained on non-zero samples, was also a classification model. As we had a limited number of samples to work with, since about half of were filtered out, only three classes were used. The boundaries of the classes were chosen so every class had a similar amount of samples from the dataset. The three classes we determined were: Agents with a CTR of less than 0.1%, more than 0.1% but less than 0.2%, and lastly more than 0.2%. This technique of using two models focused on different types of classification improved the overall accuracy of the program substantially.

To decide what kind of model would be most appropriate, we tested ten different classification models from Scikit learn. These were all run on the same training and test data to see which scored the highest accuracy. The models tested were nearest neighbour, linear support-vector machine, radial basis functions support-vector machine, Gaussian process, decision tree, random forest, multi-layer perceptron, adaptive boosting, naive Bayes and quadratic discriminant analysis. Out of these models, adaptive boosting turned out to be the most accurate.

After implementing adaptive boosting, we also tested XGBoosting. After some experimentation the result was a minimal increase in accuracy of less than one percent, however the training time was more than doubled. As such we still considered adaptive boosting the superior choice.

Adaptive boosting. Adaptive boosting, more commonly referred to as AdaBoost, is a classification model that was introduced in 1995 (Freund et al., 1999). It is a form of learning model that centers around iterating over a training set using a set of different learning models to create estimations. In the context of our thesis, we used decision trees.

After a first layer of estimations is completed, a remodelling of the training data is done. Previously evenly distributed weightings of the data is now altered so the samples incorrectly classified by the model are more significant in training for future iterations. In other words the model is designed to use a weak learner with focus on the parts of the training that are more difficult to establish rather than an even iteration over the set.

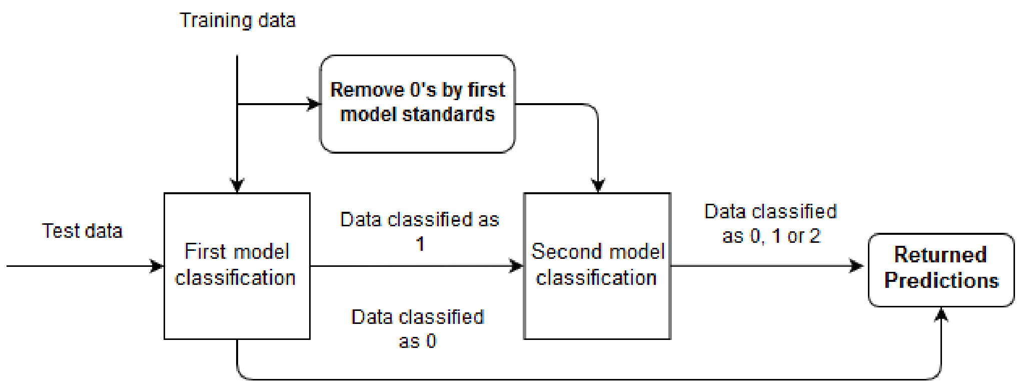

Training data
Remove 0's by first model standards
Test data
First model classification
Data classified as 1
Data classified as 0
Second model classification
Data classified as 0, 1 or 2
Returned Predictions

Figure 2.5: *A graphic representation of our complete sequence of classification models*

Specifying hyperparameters. After deciding on AdaBoost as the classifier the next step was tuning the hyperparameters. We used decision tree as a base since this was the default option in Scikit learn. This meant the hyperparameters to be determined were the depth and the number of estimators. We wrote a simple function that trained and evaluated a model and saved the results, then repeated this while incrementing the depth by one each iteration. We tested with values ranging from 1 to 20 and found that 8 was the best. Then the same process was repeated for the number of estimators, this time with values from 1 to 1000. We decided to set this to 250, even though the accuracy did continue to increase with the number of estimators. The reason for this was that after 250 the accuracy increase was trivial while the training time increased exponentially.

2.5.3 Ablation study

At the end of the project an ablation study was done. An ablation study is a procedure to evaluate the quality of the selected features of a given model.

This is conducted by recording the model accuracy for the test set, followed by removing one feature. After removing the feature, the model is re run and the new accuracy is compared to that of the complete model to see if the model becomes better or worse. The exact change is recorded, and the feature is re-inserted into the model, followed by repeating the process for another feature until all have been tested.

The result of this study can be seen in Section 3.4

2.5.4 Image-based prediction

At the core of every ad, no matter the settings of the agent, one of the most essential influences for an ads performance is the creative of the ad. The model could suggest the best settings but if the ad image is not enticing whatsoever the CTR will still be close to zero. Therefore we attempted to incorporate images as part of the input to the model. We tried two different methods, first using a *convolutional neural network (CNN)* built with Keras, and then a model built using predictions from a pre-trained network called ResNet-50.

Convolutional neural network. We built a sequential model with Keras containing three sets of convolutional and max pooling layers, followed by 2 dense layers including the output layer. The model would predict the CTR into three classes, the same classes as the model for categorical data. The predictions from the CNN and the model trained on categorical data would then be used as input to a new model. A basic type of ensemble learning. The dataset had to be restricted to agents with images that were accessible. As a result many agents that could previously be trained on were now removed due to the ad being a video or a GIF, or simply because the image could not be found. Mainly old campaigns with images that were not hosted by Emerse themselves were no longer available.

Since campaigns often consist of multiple agents, many agents use the same image. In order to avoid training on the same image multiple times, only one agent per creative was allowed. This meant that generally only one agent per campaign could be used and the training set was significantly reduced.

All of the reasons mentioned above meant there were only 1422 images to work with. These images had different shapes and the model required everyone to have the same size. So before training every image was resized to the most common size, being 300x250 pixels. Figure 2.6 shows an example of a resized image.

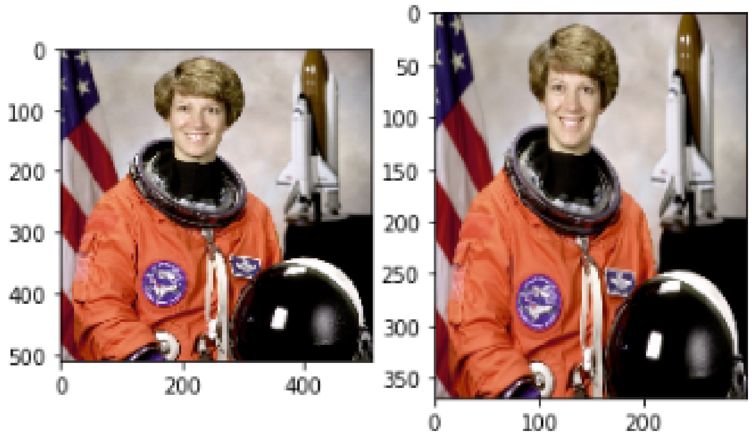


Figure 2.6: *A sample of a resized image. The picture is a stock photo of Eileen Collins taken by NASA.*

Pre-trained network, ResNet-50. In the second attempt to use images to increase accuracy, we used a pre-trained network to extract features from the image. The features could then be used as input, along with the categorical data from the agent, to a new model. We decided to use ResNet-50 that is trained on the ImageNet dataset. ResNet-50 originally returns a prediction of what object that can be seen in the image. There are 1000 possible objects that it can classify the image as, for example different kinds of animals like howler monkeys or different types of clothes like a cardigan or a bow tie. However since in this project the features of the image is whats interesting, not what kind of object is in it, we used the output tensor from the second to last layer (before the classification) instead. This tensor is 2048 features long and two dense layers was used on it before it was concatenated with the categorical data and used as input in the final classification model. An illustration of the model is shown in Figure 2.7

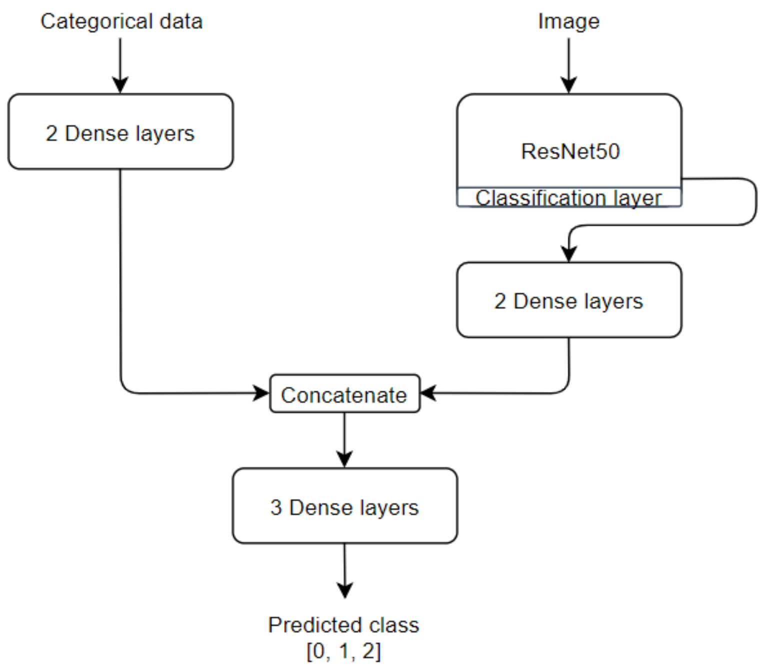


Figure 2.7: Illustration of the multi-input model with a pre-trained network (ResNet-50) for the images where the second to last layer is used as input.

The images for the pre-trained network also had to be resized. ResNet-50 has a few different acceptable sizes and we chose 224x244 pixels simply because it was the largest.

2.5.5 Ensemble learning

Ensemble learning is the method of combining multiple models in order to improve results (Chollet et al., 2015). By using different models to predict and then pool together the predictions, a more accurate final prediction can be reached. The method is based on the assumption that different independently trained models are accurate for different reasons and each of them knows part of the "truth". Hence by combining all models the whole "truth" can be seen.

According to van Rijn et al. (2018) ensembling classifiers is, in many cases, the most accurate kind of classifier available. What is known as heterogeneous ensembles are especially good. These are ensembles with models built using different techniques. For example in the context of this project an AdaBoost classifier was used along with a CNN. Other techniques could be support-vector machines or Bayesian classifiers. There are also homogeneous ensembles which involve using multiple models built with the same technique.

2.6 Utilizing the model

The second step of the project was to build a program utilizing the designed model to predict the optimal settings for a given agent. The structure of this program is as follows. First taking a given agent complete with all its selected features, including both its constants such as active time and budget, and its variables such as hour-of-week filter and targeted devices. Then iterating over the variables, trying all combinations while predicting the resulting CTR. The optimal settings could then be found among the combinations classified as the third class, representing a good CTR. The result was a program that could properly analyze an existing agent and give recommendations for what settings should be changed to increase performance.

2.6.1 Selecting between equal classes

A number of issues arose due to the exclusive usage of classification rather than regression in the model. Primarily, classification is not capable to estimate a “best value”. An immediate solution would be to select a result at random from those belonging to the most optimal class. This however raises a second problem, being the risk for false positives.

To deal with this second issue, we implemented an algorithm to instead extract the most trusted setting. The most trusted setting is decided by creating a set of *mutants* that has the same constants as the original agent, but different settings on the variables. This is done for every possible combination of those values, followed by predicting the class of every mutant. All mutants not predicted as class 2 (good CTR) were discarded.

After this is done, an *Assurance value* is estimated for each variable. This is done by looking at the sample distribution for each individual variable, comparing how many more samples the most popular value has than any other value. For example, if the optimization strategy variable has the values `off` and `on`. Perhaps there are 10,000 instances of class 2 mutants, where 6,000 have *optimization strategy* set to `on` and 4,000 has it set to `off`. As the most popular value is `on` due to having 60% of all instances, the variable *optimization strategy* is estimated to have an assurance of 60%.

This is done for all the other values, and the one that holds the highest percentage is “locked”, meaning to discard all mutants not having the chosen variable being equal to the most assured value. In this example case only the mutants with *optimization strategy* set to `on` is kept. The program then calculates the second most assured variable, followed by locking this as well. This process repeats until all variables are locked, resulting in suggesting the locked variables as the most probable successful setting for the tested agent. In the case of a tie for assurance, all remaining variables are locked for the most common value of each variable. If there is a tie between the most common value, the algorithm checks the previous iterations for a common value, and if non is found sets the value to a default setting, for example "all hours" for the hour-of-week filter. The process is visually described in Figures 2.8-2.12.

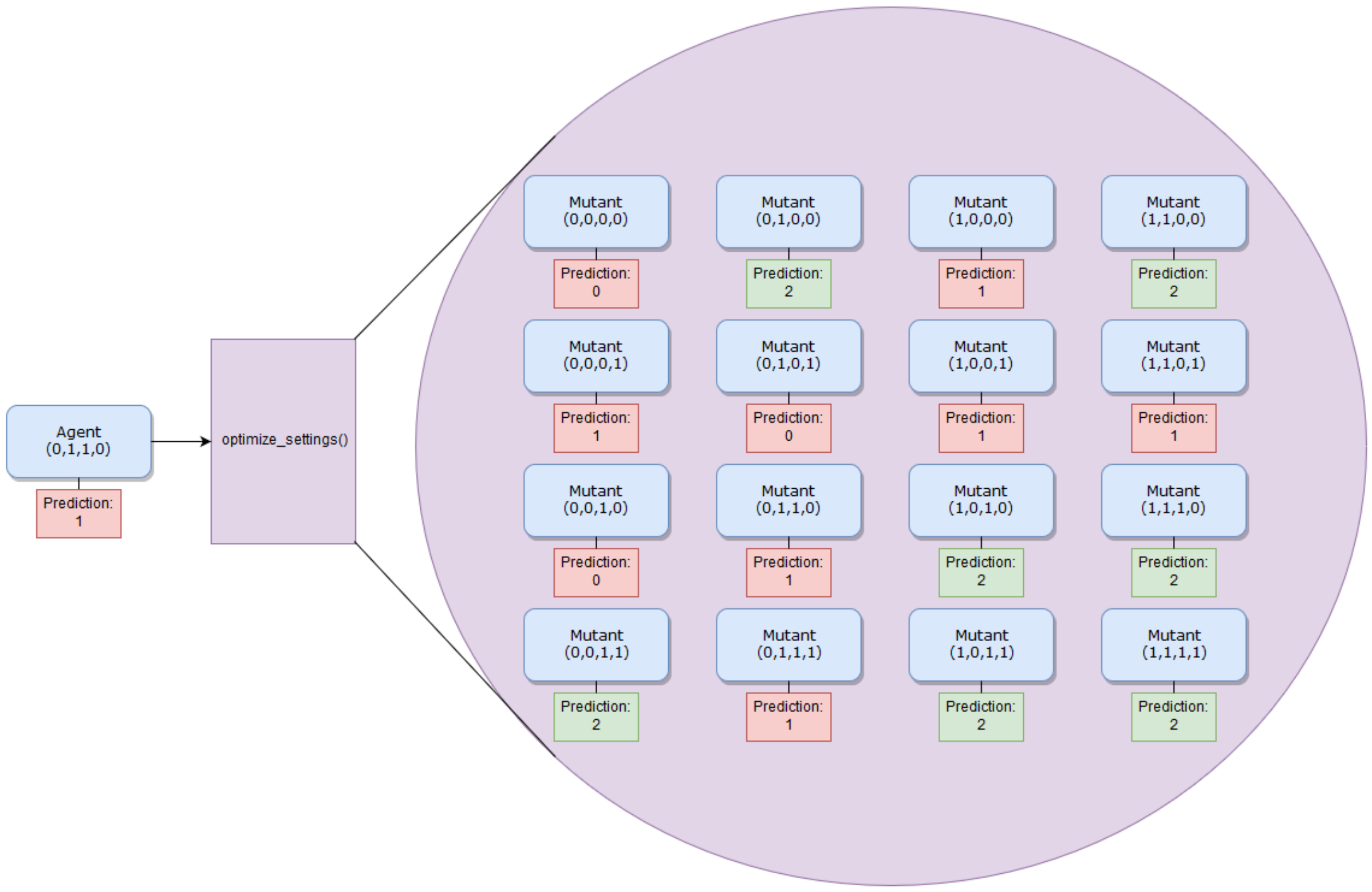
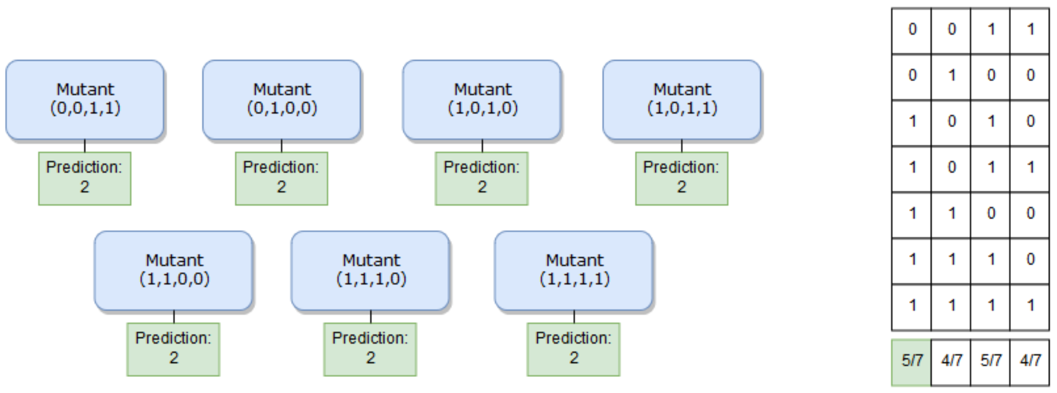


Figure 2.8: Create mutants and predict results



0	0	1	1
0	1	0	0
1	0	1	0
1	0	1	1
1	1	0	0
1	1	1	0
1	1	1	1
5/7	4/7	5/7	4/7

Figure 2.9: Extract the best mutants and select the most certain variable based on the most frequent value for each variable

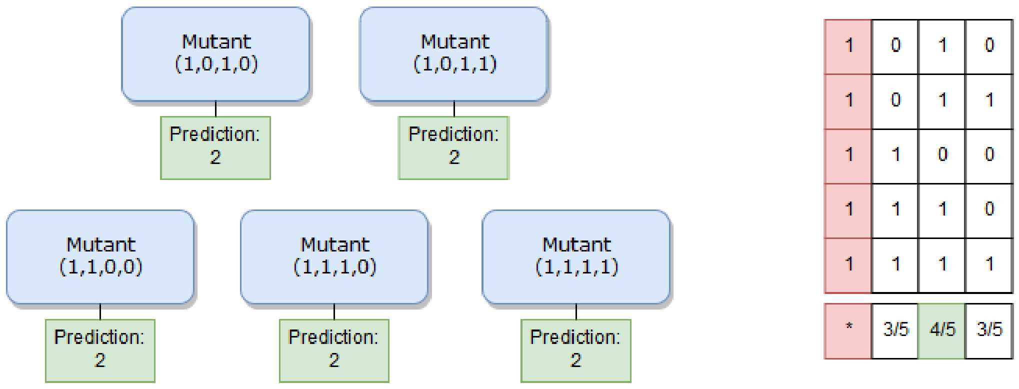


Figure 2.10: Remove all mutants with other values than that of the most certain for the selected variable and repeat

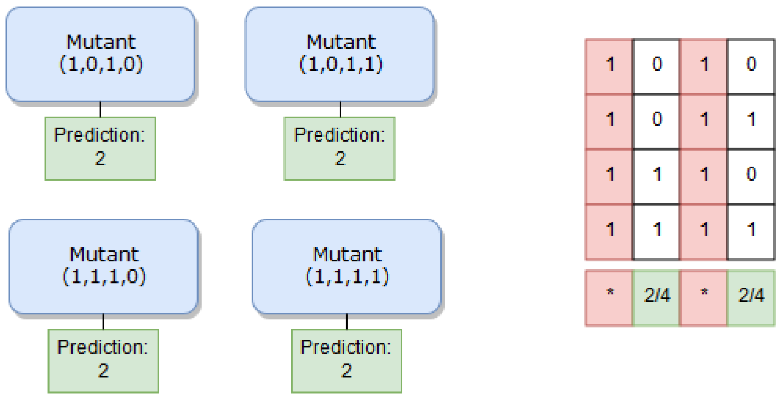


Figure 2.11: Continue until tie is found, and establish the most frequent value of the remaining variables. In the case of a tie the most frequent value in the previous iteration takes precedence.

2.6.2 Live testing

To evaluate whether the model is actually accurate, we conducted live tests. This meant generating suggestions for an actual agent buying real impressions and getting clicks from real users. We took an agent running with settings predetermined by a human operator at Emerse, and copying it into a second agent. This new agent was then given to our program which suggested new settings. Both agents were then left to run until their budget was gone. We then compared the CTR of the original agent with the CTR of our agent with suggested settings.

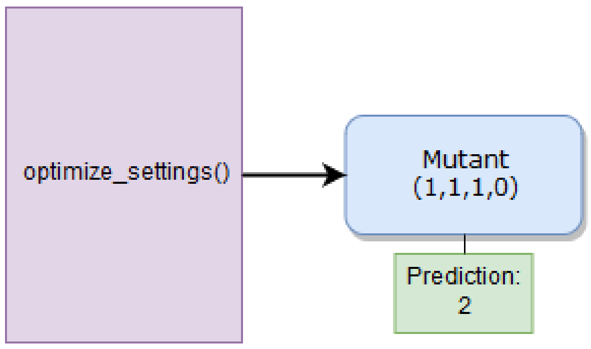


Figure 2.12: Return the last remaining mutant as the optimal settings for the agent

Chapter 3
Result

In this chapter, we will present the results of our model in training with the accuracy on the test set as well as confusion matrices. We will also show the results of the live tests.

3.1 Model accuracy

To quantify the performance of the models, we focused on the accuracy through a comparison between the predicted class of each agent in the test set and their actual class.

3.1.1 Models without images

Regression model. The initial attempt of predicting CTR, was done through linear regression. The results of this was very quickly discovered to be close to random when it came to giving predictions. Even after much tweaking of the model parameters. The graph of the result from predicting the test set can be seen in Figure 3.1. This clearly shows that the regression model was performing very poorly.

Classification model. For the first complete prototype, the model consisted of a two-step classification model using AdaBoost to estimate the result for the agents. The accuracy of these two classification resulted in 92% for the first step, and 58% for the second. Confusion matrices for both steps can be seen in Figure 3.2.

3.1.2 Classification with image-based prediction

Here we will present the results of the two different approaches to image based prediction.

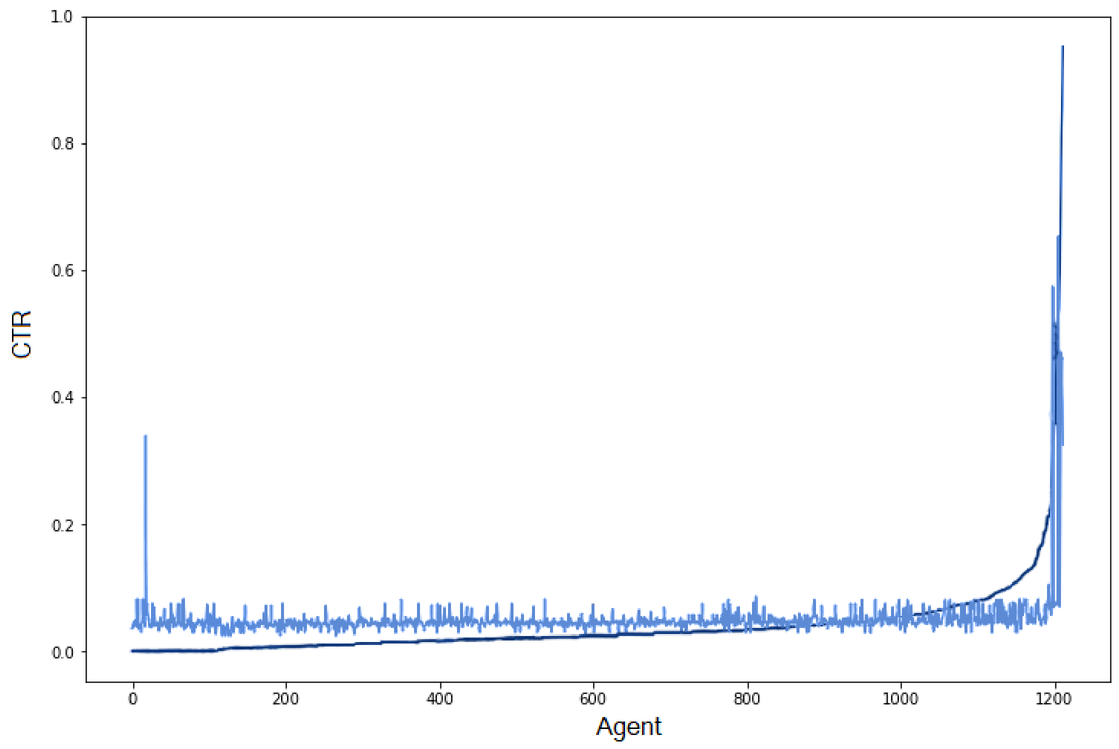


Figure 3.1: A Graph depicting the true CTR values for the regression models training set (dark blue) and the predicted CTR for the same set (light blue)

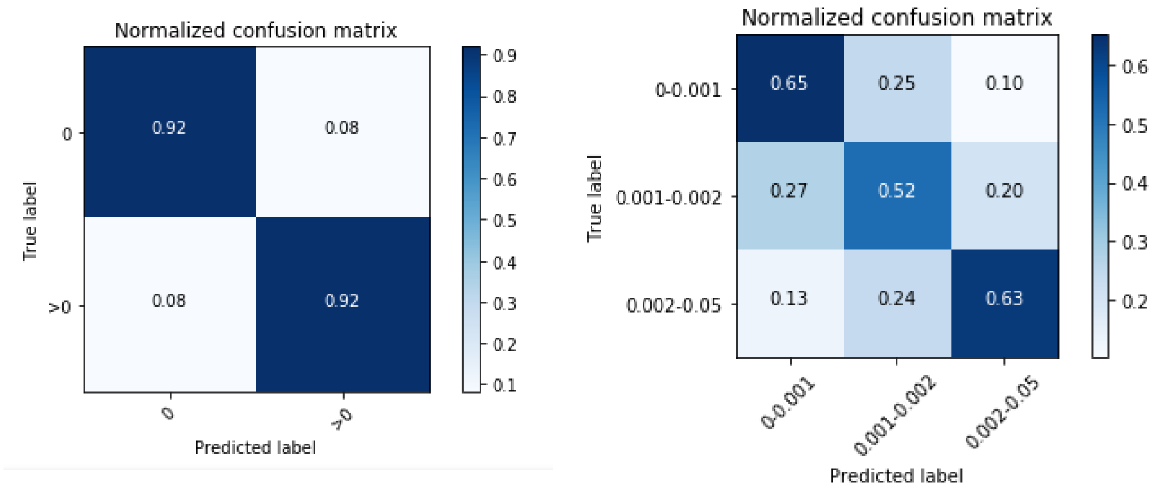


Figure 3.2: Left: Confusion matrix for the first classification step. Right: Confusion matrix for the second classification step.

First attempt (image classification on its own). Our attempt to train a CNN proved unsuccessful, as only around 30% accuracy was reached. This is worse than if the model would guess randomly between the three classes. The confusion matrix showed that the model would always predict only one of the three classes. Which of the classes was picked varied between attempts seemingly at random. In Figure 3.3 one such confusion matrix can be seen.

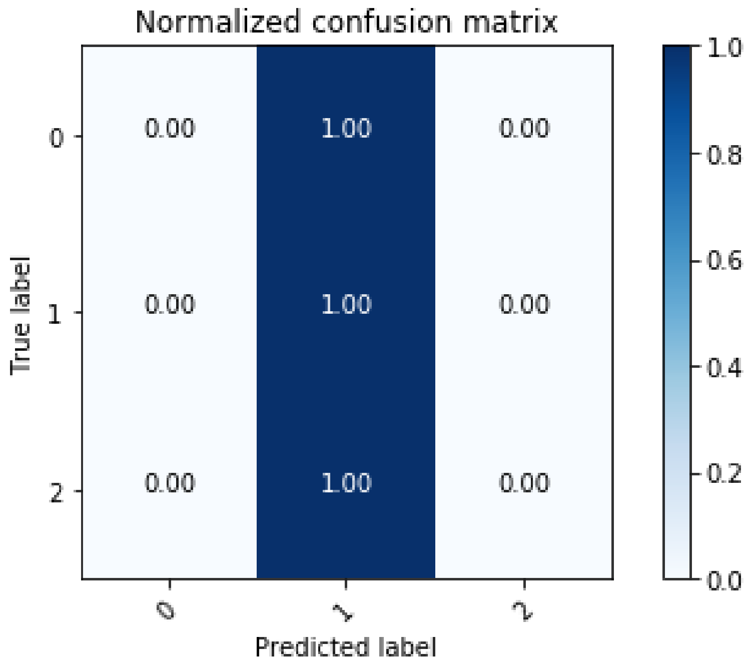


Figure 3.3: Normalized confusion matrix for our CNN.

Second attempt (pretrained network prediction). For the second iteration, we used ResNet-50. The accuracy improved markedly compared the previous attempt with a resulting accuracy of 52.8%. However, even with this improvement the multi-input model with categorical data and images was not better than the original model that only used the categorical data. The confusion matrix, Figure 3.4, for this model also shows that it was much worse at classifying agents in class 2, an important aspect in the context of this project. As a result, we decided that the final model had to remain without images.

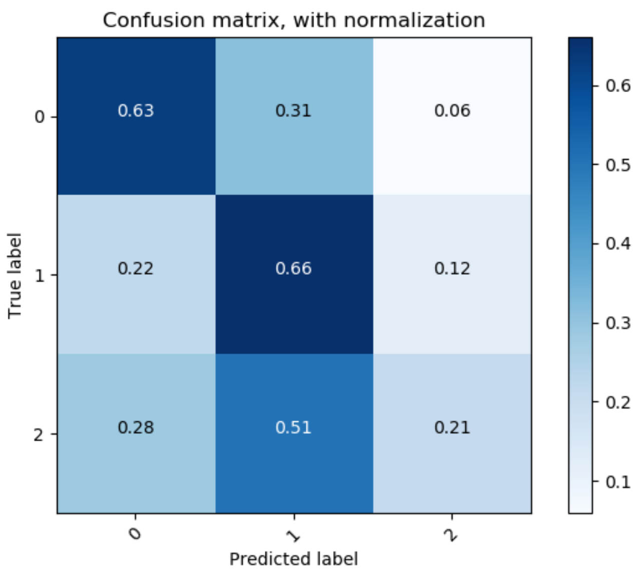


Figure 3.4: Confusion matrix for the model using images to assist prediction

3.2 Live testing

To establish the certainty behind the model and the results generated, we ran a series of live tests using real ads managed by Emerse. Their specifics and results are detailed below.

3.2.1 First test

We ran the test by using an existing ad that had a campaign budget of 62 dollars. To test this ad, we ran the original settings through the program and created a new agent with the settings suggested by the program. This new agent was then put up live with a budget of 52 dollars and was allowed to run simultaneously as the original agent. The CTR could then be compared between the two.

The ad used was an advertisement banner by a company named *Eriks fönsterputs* for cleaning windows in Sweden. The banner itself can be seen in Figure 3.5.

Test result. The test agent gave a resulting CTR of 0.0010 over 22 *unique clicks*. We compared this with the original agent, which received a CTR of 0.0007 over 19 unique clicks. This netted an increase in CTR by almost 43%, showing a clear indication that the program was capable of positively enhancing the ads performance through the given suggestions. In Figure 3.6 we have plotted the CTR of both agents each day.

A noteworthy remark is that the cost per click was also reduced greatly. This meant that the optimization was not only beneficial in terms of effectiveness but also lowering the cost of clicks, as the original had a cost per click of 2.695 USD and the test agent had 2.363 USD.

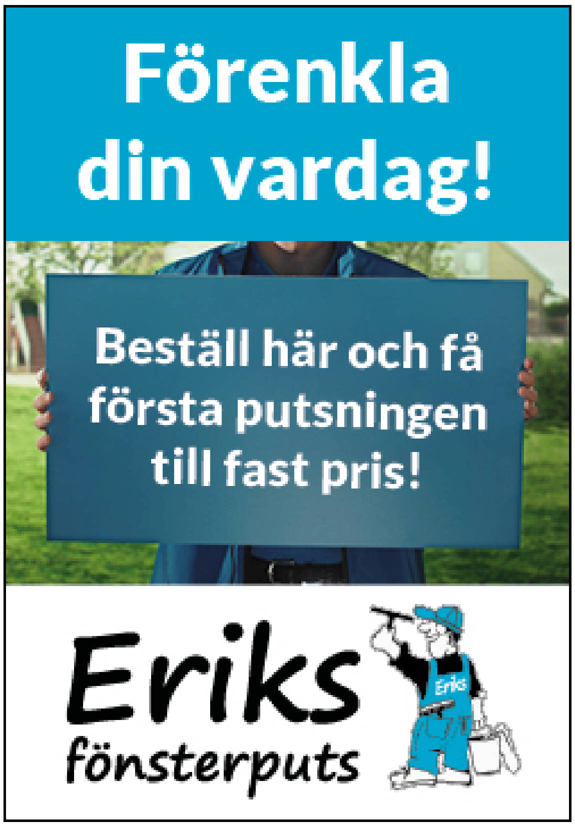


Figure 3.5: The ad banner of the first live test

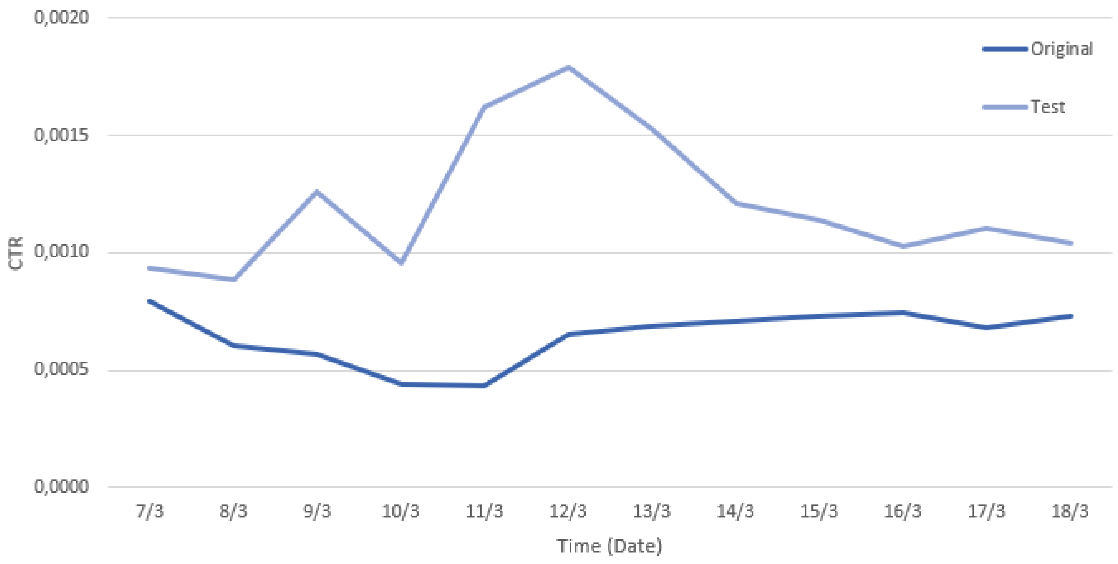


Figure 3.6: A graph over the live test depicting the CTR of the original agent and the test agent.

Confidence interval. To establish how trustworthy these results are, we calculated the confidence interval. As *unique clicks* are independent and occurs at a constant rate, the function for CTR was described as a Poisson distribution. We did this for the summed result of clicks and impressions for both agents, which can be seen in Table 3.1.

Agent	CTR	Confidence interval 90%	Confidence interval 95%
Original	0.0007	0.0005 - 0.0010	0.0004 - 0.0011
Test	0.0010	0.0007 - 0.0014	0.0006 - 0.0015

Table 3.1: Confidence interval for the first test

3.2.2 Second test

This test was done in a very similar manner to the first live test. The only real difference was that the test agent and the original agent did not run simultaneously. The original ad campaign started on March 19 and ran until April 1. The test agent was started on March 25 and was kept running until April 10. The budget was also larger for this campaign with a budget of 112 dollars.

As with the previous test, the original parameters for an agent were run through the program, and a new agent was created with the parameters suggested. Following this we set up the new agent in the DSP with a smaller budget of 102 dollars.

This test used an ad banner for a company named Phonak, advertising hearing aids. The banner for the ad that was used for the test can be seen in Figure 3.7.

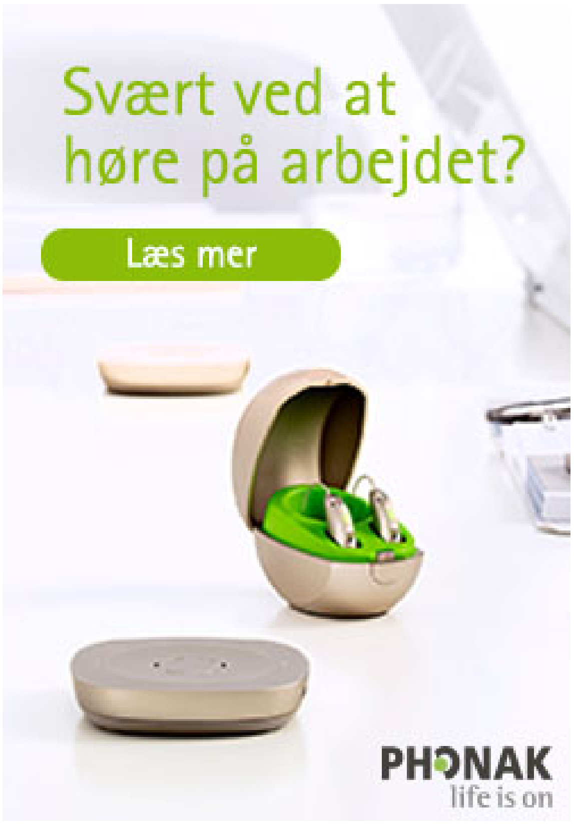


Figure 3.7: The ad banner used in the second live test.

Test result. This second test gave a poorer result than the first, having a CTR of 0.0006. When compared to the original agents CTR of 0.0008, this is a decrease of 25%. The results from the test is displayed in Figure 3.8.

The amounted cost for the advertisement also suffered as the cost per click was increased from 2.50 USD to 3.45 USD.

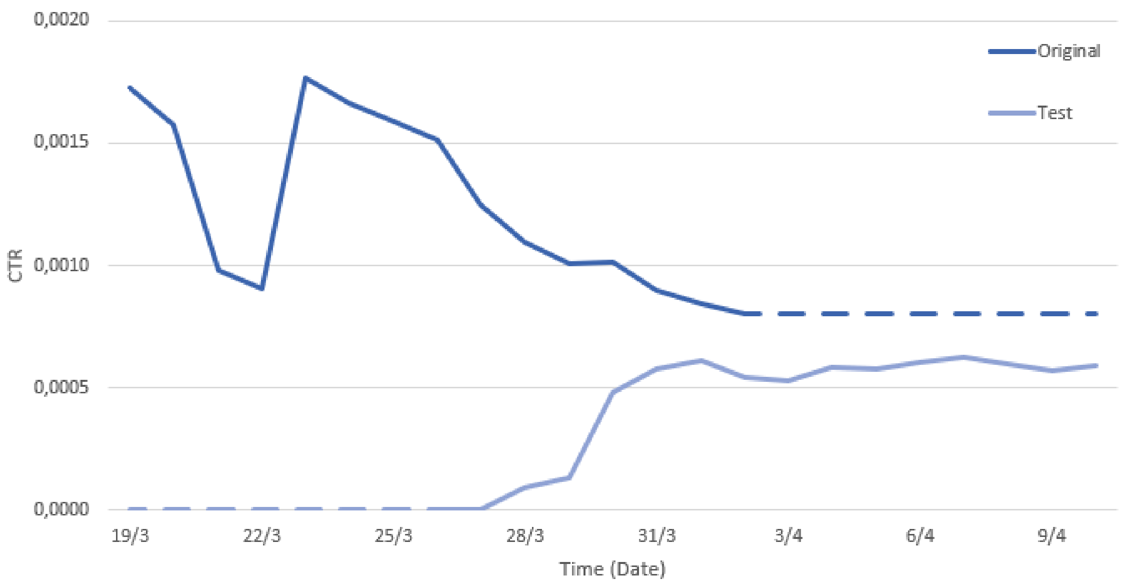


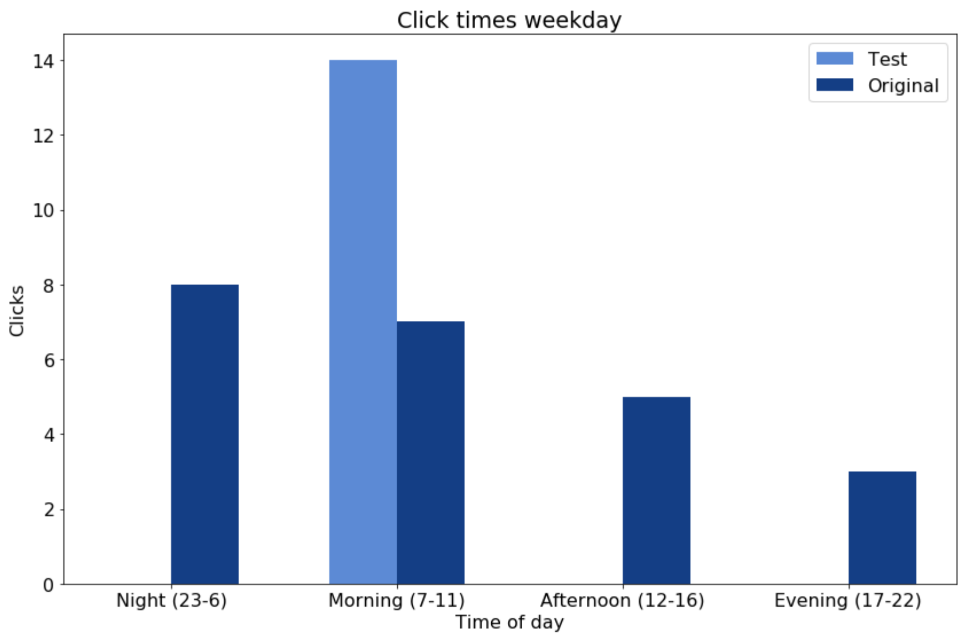
Figure 3.8: A graph over the second live test depicting the CTR of the original agent and the test agent.

For this test, we looked closer on when clicks occurred to assess the hour-of-week filter. Our model suggested quite a specific hour-of-week filter, only between 7 and 11 on weekdays and all hours except between 7 and 11 on weekends. The original agent was active during all hours. In Figure 3.9 the amount of clicks throughout the day is shown. The light blue graph is the test agent with settings chosen by our model. If we use the original agent as a reference for when the ad received clicks, we can evaluate the effectiveness of the suggested hour-of-week filter. The diagrams show that the test agent chose one of the most popular times during the weekdays. However on the weekends it chose not to run in the mornings, which turned out to be the second best time for the original agent. It was also active in the evenings on weekends where neither of the agents got any clicks.

Confidence interval. Through the same method as for the first test, the confidence interval is estimated in Table 3.2.

Agent	CTR	Confidence interval 90%	Confidence interval 95%
Original	0.0008	0.0006 - 0.0010	0.0006 - 0.0011
Test	0.0006	0.0004 - 0.0008	0.0004 - 0.0008

Table 3.2: Confidence interval for the second test.



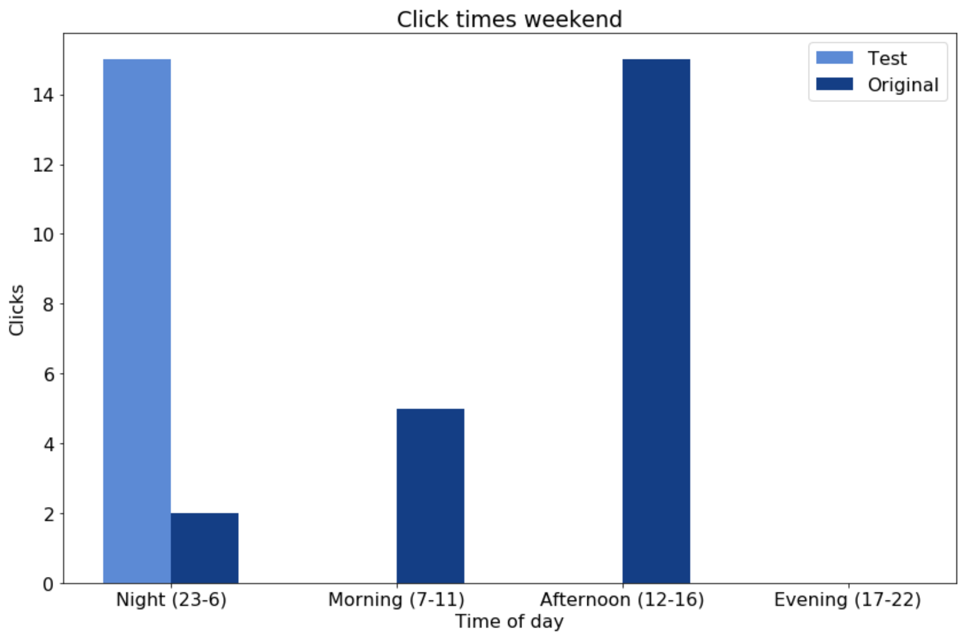


Figure 3.9: Amount of clicks per hour for the second test. The test agent was active for mornings during the weekday (above) and for all hours except mornings for the weekend (below).

3.2.3 Continued testing

After the two initial large-scale tests, a larger batch of six agents where run simultaneously in the same manner as the previous two tests, but with smaller budgets. Out of the six agents, four had budgets of 20 dollars each allocated to them and two had 10 dollars.

All tests were run until the budget ran out, followed by a CTR comparison.

Statistical summary. In Table 3.3 the results of all live tests are shown. Each row represents a pair of agents, the test agent and the original agent.

Test			Original		
Unique clicks	Impressions	CTR	Unique Clicks	Impressions	CTR
4	3172	**0.0013**	3	6447	0.0005
3	8410	0.0004	**6**	10,771	**0.0006**
2	6964	0.0003	**13**	11,051	**0.0012**
11	8107	0.0014	11	6596	**0.0017**
28	51,260	0.0006	**45**	53,421	**0.0008**
88	42,132	0.0021	72	34,896	0.0021
4	1154	**0.0035**	**16**	7425	0.0022
22	21,192	**0.0010**	**19**	26,087	0.0007
Total clicks	Total Impressions	Average CTR	Total clicks	Total Impressions	Average CTR
162	142,391	**0.0013**	**185**	156,694	0.0012

Table 3.3: Test results for every test we ran. Each row represents one test agent and original agent pair.

3.3 Baseline test

Aside from running tests comparing our program to a human operator, we also ran a test comparing our program to a baseline. In this case our baseline was created by randomizing parameters for 5 agents with a budget of 10 dollars. We then compared the resulting CTR from these tests to an agent created with suggested parameters from our program, this agent had a larger budget of 50 dollars. The individual results of each agent can be seen in Table 3.4.

The different agents with randomized settings had varying results. For a better comparison to the agent with the suggested setting, we took the average CTR of the randomized agents. The average CTR of these was 0.0010.

Clicks	Impressions	CTR
Random settings		
0	14	0
2	2,752	0.0007
1	2,099	0.0005
6	2,874	0.0021
1	2,723	0.0004
Predicted settings		
14	11,904	0.0012

Table 3.4: Results of the baseline test

3.4 Ablation study

In Table 3.5 the result of an ablation study is presented for all features used in the final version of the model for both the first and second step. Ablation studies is a statistically sound method for establishing a ranking of significance for the model features (Meyes et al., 2019).

In Table 3.5, the ablation score is defined as the percentage of accuracy lost when the specified variable is removed from the model. A negative score means the model benefited from losing the feature.

Baseline	Accuracy first step	Accuracy second step
Unmodified	0.921915	0.575781
Feature name	Ablation score first step	Ablation value second step
IAB category	0.03794	0.014844
Flight time	0.03130	0.014844
Optimization Strategy	0.00147	0.005469
Budget	0.02320	0.001562
Device type	0.00110	0.000781
Is Video	-0.002578	-0.002344
Width	-0.006262	-0.004688
Pacing type	0.00405	-0.007031
Height	-0.000737	-0.008594
H/W ratio	-0.000000	-0.009375
Hour-of-week filter	0.00037	-0.010157

Table 3.5: Ablation study result for all features in both steps of the final version of model.

3.4.1 Model output after ablation

After experimenting with the features in regard to the ablation study, we attempted to gain an optimal accuracy value for the model. We removed the most negative feature according to the ablation study. Then we made a new ablation study to determine the next feature to remove and so on until no negative features remained. This however, meant removing variables from the model, thus making it unusable in practise due to not being able to optimize over the agents.

The result of this was an agent with an accuracy of 59,5%, that consisted of the same features as the unmodified agent excluding the features *Hour-of-week filter*, *Height*, *H/W ratio* and *Is Video*. The confusion matrix for this model can be seen in Figure 3.10

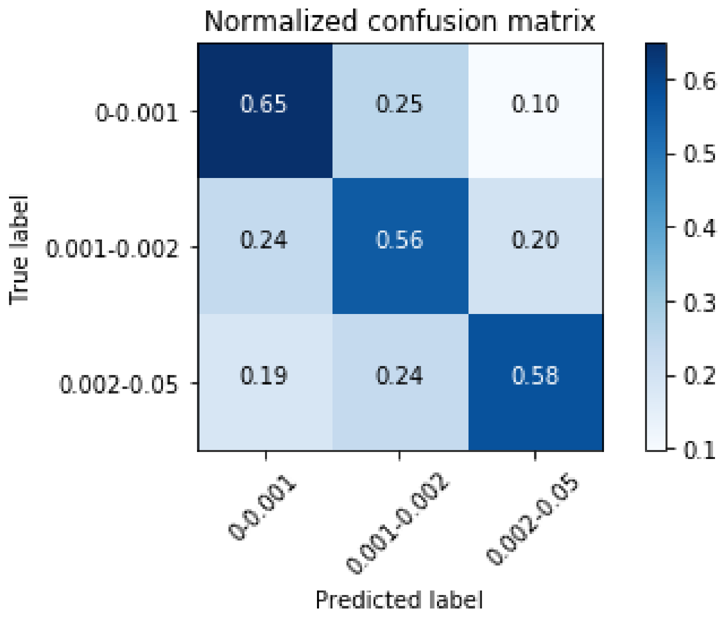


Figure 3.10: Confusion matrix for the model with features removed based on the ablation study

Chapter 4
Discussion

In this chapter, we will discuss some potential error sources, analyze the result, and reflect on the effect the data size limitations had on the project. We will also discuss some attempts at improving the model, and what could be done in the future.

4.1 Result analysis

Here we will review the results we documented in the previous chapter further. We will discuss the reasons for why we got these results and also explore possible error sources.

4.1.1 Error sources

Test evaluation. One source of error, is that for all the tests, the original agent had been running for some time before each test agent was set up. This meant that we were forced to limit the comparison between days when both agents were active. The reason for this was to avoid the result being incorrectly worsened, due to CTR generally being harder to increase the longer an ad has been running. This is based on the notion that the group of potential clickers for an ad decreases over time, as people whom have seen the ad have already made the choice to click or not. This trend can also be seen clearly in both the Figures 3.6 and 3.8, where the CTR increases for the first few days and then slowly decreases. In practice however, this has very little impact on the results as we disregarded all data from before both agents were running.

This wouldn't be a big problem if the target audience for an ad is large. However, in the case of the second test, the target audience was Danish speakers in need of hearing aids, which is a quite niched group. As a result the total amount of clicks for both agents was very low, and thus the result became somewhat uncertain.

Another source of error is the fact that most tests ran on a very small budget, resulting in very few clicks. The result of this is that the confidence intervals for both the first and second tests did not give clear results. This leaves some uncertainties whether the results were a stroke of luck, or due to our program actually having impact.

One of the reasons for the test agents resulting in such a poor performance can in part have been caused by the hour-of-week filter. In second test, depicted in Figure 3.9, we can see that the accuracy of the program when deciding the optimal times to run, is very varying depending on the the weekends and weekdays.

For the weekday, the model did well in selecting the optimal time to display ads. However during the weekends, the selection was very poor as both afternoons and evenings were the chosen times to air on even though these resulted in no clicks for the agent.

We think the poor performance of the hour-of-week filter was in part affected by our way of scaling this feature. If the agent had at least one active hour in the time span then it would count as being active for all of that period. For example if an agent was just active at 8.00 then it would still count as being active the whole morning period, 7 to 11. It may have been better to base the decision on the average number of active hours for that period instead.

Baseline test. The results of the baseline test were not as decisive as we thought they would be. The CTR of the agent with settings suggested by our program was only 20% better than the average CTR of the randomized agents. The reason for why the CTR was similar was because one of the randomized agents performed really well. Four of the five randomized agents had a low CTR, but one of them had a much higher CTR of 0.0021. Looking at the settings for this agent they are not much different from the others. The most unique setting is the hour-of-week filter which says the agent is only active during the night on weekdays and during the afternoon, evening and night on weekends. There is no clear reason for why this randomized agent performed much better than the other ones. We believe this is an anomaly, and this is the reason for why we created 5 different randomized agents instead of just one. If this agent is ignored however, the average CTR of the randomized agents would be 0.0005. A significantly worse performance.

4.1.2 Ablation study

We analyzed the result of the ablation study. It appeared that the removal of most individual features had a very small impact on the model accuracy, with an ablation score smaller than 1% in all but three values.

However, a few features were still found to have much individual importance. The most impacting features according to the results were IAB category and flight time, i.e the duration of the campaign. The most impacting feature in terms of improving the model when removing the feature was the hour-of-week filter.

For IAB category and flight time it seemed natural that these values would have a large positive contribution to the predictions. The IAB category of course dictates the target audience, which can differ greatly in both device use and active hours during the day. Flight time as well can have great impact. The group of potential clickers can be exhausted if the flight time is set too long. If an ad continues to run after this exhaustion, the amount of impression will increase but the clicks will not. Users who have already seen the ad and clicked on it are less likely to click on it again. In the end this leads to a poorer CTR.

However, hour-of week-filter was not expected to have such a negative contribution to the model accuracy. Mainly because of the reasons it was implemented in the first place, as explained in the method.

The reason for why hour-of-week filter was a poor feature is most likely because of the lack of data. This prevented the model from learning good enough patterns. A big issue was that a lot of agents trained on did not use a specific hour-of-week filter, i.e they were active all hours. With more data, and specifically more samples with a varying hour-of-week filters would likely yield a better result.

It was a surprise that the budget feature had such a low impact on the model. We expected this feature to be contributing more. Both due to experience shared from employees at Emerse, but also as it was shown in previous non-tracked experimentation's with feature removal that it had a very large impact when the number of features was small. This is likely due to correlations with other variables, such as flight time that tends to scale with the budget in at least some manner.

Even though Pacing type might contribute negatively to the model, it is most likely required. The reason for this is that the model has to have a pacing type either way, as there is no clear default value.

It is unfortunate that over half of the values included in the model had a negative score, meaning that the model would have performed better without at least one of these features. As such, potential future work would be to apply the results of this study and run new tests. If that would have been the case, then the test might have resulted in a higher performance for the test agents.

However, it is worth mentioning that the accuracy does not actually benefit from removing all of these features at once, as we showed in the Results. The result of the ablation study is not as bad as one might get the impression of when initially looking at Table 3.5.

4.1.3 Regression

As can be seen in the result chapter and the Figure 3.1, it is clear that the regression model was far too inaccurate to be used in any meaningful way. We believe that regression is viable but it requires more data. Looking at other papers attempts at applying regression to predict the CTR of individual viewings of ads, it is clear that with more specific data and more variables to describe the environment in which the ad is placed, it is likely that regression may yet have a better potential (Guo et al., 2018).

4.1.4 Classification

Analyzing the results given from the first test, it would initially seem that the model is performing excellently. However, due to the confidence interval and the tests that followed, it would rather seem that the model in its current state is relatively unstable.

Given the statistical summary of the tests however, the average increase of CTR was positive. This meant that the model in fact does perform slightly better on average than a human operator. Taking this into account along with the fact that this model has room for improvement, it would seem clear that this method of CTR optimization can be very beneficial if work is continued to increase performance further.

It is important however, to consider the fact that the model does not evaluate the best settings. Instead, the model selects the safest option among those that are believed to be among the best. This means that the model is prone to reach a ceiling for improvement, unless the model is altered to either go back to regression, or create models with more classes to gauge success. This would of course first require a big increase in model accuracy, as more classes are very likely to result in an accuracy so low that the result would suffer. This is due to difficulties separating the classes, even if the attempted predictions would be more specific.

4.2 The importance of abundant new data

Throughout the entirety of the project, the amount of data available to us for training was a constant problem holding progress back. This problem heavily relates to how difficult it is to produce new data. In our case, data is created slowly through new customers approaching the ad management company. As such it is not reasonable to expect new data to be produced and collected throughout the project. This means that the data available to us from the start is the data used throughout the project.

The largest setback this caused can be seen in the classification attempts with images. Multiple banners did not fit into the model, due to being GIF's or videos rather than regular images. This along with many images being unavailable resulted in the very small batch of 686 images after filtering out agents with a CTR of zero. We quickly realized that this was far too small of a dataset as the model was so prone to over-fitting that no efficient and accurate output could be generated and used.

It is clear through this that no matter how well the model is designed and enhanced, more data is always helpful to make proper estimations. It can also be said that this project, if continued, would most likely improve incrementally with time, as data is accumulated. Alternatively this would also be the case in another environment where more data is already available.

A big aspect of this, is the relevance factor. What is meant by this is that the advertisement industry is a very fast-paced market in terms of changing both tools and strategies (Johnston, 2008). As a result of this, many agents that were included in the training of the model in this thesis might not have been as relevant as others. As the strategies to optimize CTR constantly changes with the times due to both ads behaving differently, and people developing a changed perception of the ads shown to them (Johnston, 2008).

This notion also heavily implies that the model created and trained in this thesis might not be relevant in a short notion of time, and as such need to be trained with new data. This might not be an issue assuming that more data is created at the same pace that old data becomes irrelevant. It does however mean that for the amount of available relevant data to increase, the market needs to grow in order to produce new ad campaigns faster than old ones become irrelevant. An interesting solution to this would be a larger collaboration between marketing companies, which might result in a great increase to performance for all involved.

However according to Silverman and Gaffney (2018) the advertisement market is currently increasing rapidly, as we described in the introduction, which speaks for that the model tested in this thesis has great potential for the future.

4.3 The potential for image-based CTR prediction

In theory it seemed obvious to us that introducing images as input to the model would improve accuracy. Where the ad is shown, at what times and to what people are all important aspects but the image shown to the consumer is undoubtedly a major factor towards whether the ad is clicked on or not. There are also articles of previous work regarding predicting CTR with images, one of them being *Deep CTR Prediction in Display Advertising* by Chen et al. (2016). This further proved that introducing images to the model was a good idea.

Yet when we implemented a multi-input model, with input both from categorical data from the agent and the corresponding ad image, the results worsened. Even when just the height, width and ratio of the available images where passed to the model this had either a marginal or negative impact on accuracy. There are a few reasons behind this failure which will be discussed below.

4.3.1 Resizing

Firstly one of the reasons could be that the images had to be resized before they were passed along to the model. In the best attempt with the pre-trained network, all images were resized to 224x224. This meant that the images were stretched and squeezed. Although some images with similar height and width were barely affected by the resize, there were also images with dimensions like 120x600 with text that became unreadable after resizing.

There were some ideas on how the issue with resizing could be countered, mainly by cropping the image instead of resizing it. This would mean focusing on the central part of the image. The problem with this solution would be that the crop would have to be of the same size as the smallest image, resulting in an extremely small sample taken from the larger pictures. This could probably have been solved by training large and small images separately, however due to the model already being affected by the data set being too small, it would unlikely have helped with the currently available batch of images.

4.3.2 Data batch size

Although resizing was a problem in some of the images, one major reason for why the model could not achieve better prediction accuracy, especially when training the CNN, was the insufficiency of samples. As mentioned previously, the amount of images available was limited due to a number of factors. We concluded after multiple attempts at tweaking the image based classifier, that unfortunately we had too few samples for it to learn anything usable.

Only 686 images are available to train on after the first part of the program filters out roughly half of the agents, the ones with a CTR of zero. Because of this, the model was prone to over-fitting. As such, it ended up giving a worse accuracy than predicting the CTR without images, only using the categorical data from the agent settings.

4.3.3 Pre-trained networks

With the pre-trained network the accuracy was significantly better. This was due to the simple reason that the model did not have to train on images, so the small sample size was not as much of a problem. The dataset was not as limited because all agents belonging to the same campaign using the same advertisement banner was now usable.

When training the CNN, training on the same image multiple times was avoided. However, in the new model the image features were extracted and concatenated with other data. The images were not trained on directly. The sample size was however a bit smaller compared to training without images when ads with other formats than images were allowed.

There was still a problem with the results of the model using images even though the accuracy was quite close to our best image-free model. As we mentioned in the results, and is shown in Figure 3.4, most of this accuracy comes from correctly categorizing agents as class '0' and '1'. Although this theoretically gives a higher accuracy, the practical application of our problem only handles whether an agent is a class '2' or not, representing a good CTR. As only 21% of the agents labeled '2' in the test set were classified correctly, this would mean that if this model was to be used in a live test, it's very likely that a huge portion of the agents with a genuine good CTR would be discarded as poor alternatives. As such the suggested settings for the live tests run on this model would not be particularly accurate and would probably yield a poor CTR.

4.3.4 Practical analysis

In conclusion, training on images along with categorical data is potentially a good idea. Thinking about it intuitively, images have a huge impact on the performance of an agent and as such it should be beneficial to train on the images to increase the accuracy of the predictions.

Previous work done by Chen et al. (2016) shows that it can be done. However as discovered in this thesis, it requires plentiful data to work. Chen et al. (2016) had a data set with a size of 59 million, with 118,960 unique images. Our data set consisted of around 1000 unique images, which is not enough to train a CNN.

4.4 Other attempts at increasing CTR

There were more attempts at introducing other features to the model. To involve the images more, we tried to include whether the ad banner was large or not. To define a large image we used the guidelines from IAB, which we described in the Introduction chapter. Unfortunately, this feature affected the accuracy of the model negatively. We therefore decided not to include it in the final version of the model.

Another feature we experimented with was a frequency limit for how many times a person was allowed to see an ad. This could be limited in different ways such as per day or per month. Unfortunately, only a third of the available agents used this feature. The result of this was that the feature only confused the model. However, it is likely that limiting a user to a certain number of viewings could in many cases lead to a higher CTR. This is true since a person who has seen the same ad twenty times is probably less likely to click on it compared to someone who hasn't seen it. The money spent on that impression could be spent on a new user instead.

4.5 Practical application

With the results gained from the project, it can be said that using machine learning methods to improve upon targeted ads have great potential. However, the question remains whether it can be applied in a realistic manner over time in a way so that it can be beneficial for the industry at large, or at least the company utilizing the method.

Due to the results produced, the simplicity of modifying the model built, the constantly more or less automatic growth in data, and the potential for further modifications and increases in variable usage, this method certainly has great potential. This would be true even if the model is currently lacking in dependability. It will however most likely take a long time before a model like this could be made completely autonomous and replace a human operator. The tests run during the lifetime of this project, are not enough to establish whether the program is reliable over a longer time. However due to this, it is probably much more useful as a support tool for humans managing online advertisement than it would be as an independent and autonomous tool.

It remains to see what results could have been achieved with further testing if the ablation study and other corrections would have been applied in time. It is likely that the increased accuracy of the model would have led to a better result, although it outlives the scope of this thesis.

4.6 Future work

Here we discuss some improvement to the model that could have been done if more resources and time was available

General improvements. Looking at what more can be done for the model, future work could possibly involve building a more complex model that can make suggestions to even more parameters. Another improvement is making the model capable of autonomously setting up the agents in the DSP. This would eliminate the need for the human operator, and perhaps increase the accuracy and performance even further. This will however require more available data, and as such time will have to tell if this is possible in the near future as the market grows.

Real time prediction based suggestions. Another interesting aspect not covered in this thesis, is the strategy observed and used by Emerse employee's managing the advertisement. There, an agent is set up and then repeatedly monitored for performance. This allows for tweaking the agent if the performance does not live up to the standards.

The program created in this report only evaluates the CTR predicted to be achieved at the end of a campaign. However, should the model be capable of continuously adapting to its own mistakes as the agent begins to run and live data is supplied, a higher end value might very likely be achieved.

Image classification. We think that an image classifier has great potential if enough data is provided. This is based on the fact that it's the only thing the targeted user actually sees. Considering how the model after much work achieved an accuracy only slightly worse than the one without images, this would probably surpass the image-less model if more were supplied, or more advanced models of image classification were used.

Another future improvement related to the image classification is handling GIF's and videos. This form of creative representation was very common, but had to be removed for training on images only. However, it could be interesting to try different approaches, including extracting snapshots from the videos and GIF's to include them with the image analysis. Another solution to generate more data for the images would be to slightly manipulate the images by duplicating and rotating them. This would result in similar but newer data-points for the model to learn from, perhaps improving the accuracy.

Chapter 5
Conclusion

In this thesis we tackled the problem of trying to predict the CTR in order to automate online advertisement by using a multitude of different methods. The methods used were done by collecting data, sampled over a long period of time by the company Emerse. This was followed by using this data to train machine learning models, intended to suggest the settings of an autonomous agent bidding on impressions online. The different models we tried include linear regression, classification and image-based classification (both with and without pre-trained models).

Our main difficulty in this project was the amount of data available to train the models on. This was the suspected reason for why both image classification and linear regression turning out very poorly in terms of accuracy. A model built for classifying whether a bidding agent was to receive any clicks received an accuracy of 92%, and a model for classifying agents between good, mediocre and poor results had an accuracy of 58%. The other methods received poorer accuracy, such as the image classification receiving 53% accuracy at best for selecting between the same three classes as the model not using images.

The classification models that yielded good accuracy were used to conduct a number of live tests to see if they could perform better than a manually created agent. The summary of these tests resulted in the test agents receiving an average of 0.0013 CTR. The compared agents set up by humans working with this form of ad-management, gained an average CTR of 0.0012.

Due to these results and the accuracy of the models, it was concluded that there is a big potential for machine learning when it comes to managing bidding agents. However, the result in this thesis is limited due to the difficulty of gathering data, both due to old data becoming irrelevant over time and data being collected relatively slowly. As such, even if the results gained in this thesis is satisfactory, there is a need for a growth in the industry or collaborations between companies to gain proper and useful models that could guarantee a good result in this field.

Bibliography

AerServ (2018). List of IAB categories. https://support.aerserv.com/hc/en-us/articles/207148516-List-of-IAB-Categories. Accessed: 2019-04-04.

Chen, J., Sun, B., Li, H., Lu, H., and Hua, X. (2016). Deep CTR prediction in display advertising. *CoRR*, abs/1609.06018.

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

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Fei-Fei, L., Li, K., Russakovsky, O., Mellon, C., Krause, J., Deng, J., and Berg, A. (2019). Imagenet. http://www.image-net.org. Accessed: 2019-04-11.

Francois, C. (2017). Deep learning with python.

Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612.

Graepel, T., Candela, J., Borchert, T., and Herbrich, R. (2010). Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, pages 13–20, Microsoft Research Ltd.

Guo, L., Ye, H., Su, W., Liu, H., Sun, K., and Xiang, H. (2018). Visualizing and understanding deep neural networks in ctr prediction. *SIGIR*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Interactive Advertising Bureau (2019). Our story. https://www.iab.com/our-story/. Accessed: 2019-05-06.

Johnston, A. (2008). The relevance factor. *DMNews.com*. Accessed: 2019-04-16.

Kaushik, R. V., Raghu, R., Reddy, L. M., Prasad, A., and Prasanna, S. (2017). Ad analysis using machine learning: Classifying and recommending advertisements for a given category of videos, using machine learning. In *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pages 2434–2437.

Louth, J. D. (1966). The changing face of marketing. *McKinsey Quarterly*.

Meyes, R., Lu, M., de Puiseau, C. W., and Meisen, T. (2019). Ablation studies in artificial neural networks. *CoRR*, abs/1901.08644.

MRC (2014). MRC viewable ad impression measurement guidelines.

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

Puffett, M.-C. (2018). IAB Europe report: European programmatic market sizing 2017. Report from IAB Europe with numbers related to growth and revenue as well as statistics for ads on different devices.

Seker, S. E. (2016). Real life machine learning case on mobile advertisement: A set of real-life machine learning problems and solutions for mobile advertisement. In *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 520–524.

Silverman, D. and Gaffney, B. (2018). Iab internet advertising revenue report. *Interactive Advertising Bureau. New York*, page 26. Report from IAB about the numbers related to online advertising, such as revenue and growth. Is a biannual report.

van Rijn, J. N., Holmes, G., Pfahringer, B., and Vanschoren, J. (2018). The online performance estimation framework: heterogeneous ensemble learning for data streams. *Machine Learning*, 107(1):149–176.

Yuan, S., Wang, J., and Zhao, X. (2013). Real-time bidding for online advertising: measurement and analysis. In *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*, page 3. ACM.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2019-06-11

EXAMENSARBETE Managing Programmatic Advertising Using Machine Learning
STUDENTER Carl Dahl, Pontus Ericsson
HANDLEDARE Pierre Nugues (LTH), Carl-Johan Grund (Emerse), Rasmus Larsson (Emerse)
EXAMINATOR Jacek Malec (LTH)

Hur maskininlärning kan öka antalet klick på internetreklamer

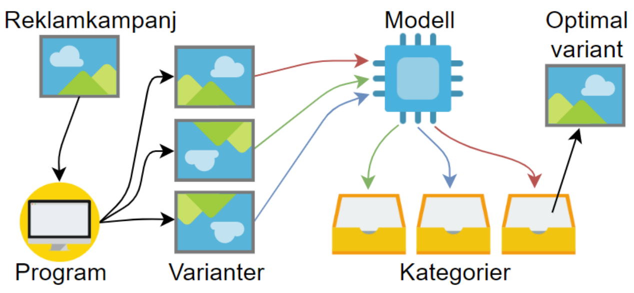
POPULÄRVETENSKAPLIG SAMMANFATTNING **Carl Dahl, Pontus Ericsson**

Marknadsföring på internet är en snabbt växande industri med många miljarder dollar investerade. Med den framskridande utvecklingen av att applicera maskininlärning på verkliga problem har det blivit en naturligt följd att försöka kombinera dessa.

I vårt projekt har vi tränat en modell som kan förutspå procentchansen att internetreklam blir klickad på. Modellen är byggd med algoritmen AdaBoost och har tränats på data från tidigare reklamkampanjer på företaget Emerse Sverige AB. Datan innehåller olika inställningar på kampanjen och även vilken klickchans den hade. De olika inställningarna är exempelvis vilka tider reklamen är aktiv, hur stor budget den har, vilken målgruppen är med mera. Modellen kategoriserade en kampanj till en av tre kategorier: Låg, medel och hög klickchans. Modellen gissade rätt i 58% av fallen.

Vi använde sedan denna modell för att försöka ta fram de optimala inställningarna för en reklam så att den visas med de förutsättningar som ger så hög klickchans som möjligt. Detta gjordes genom att ge modellen en kampanj och förutspå dess klickchans. Sedan ändrade vi på värden i kampanjen och gav den till modellen igen. Genom att testa alla kombinationer av inställnigarna så kunde vi hitta de optimala.

Programmet testades med åtta tester på riktiga reklamer där vi jämförde klickchansen på kampanjer som vårt program ställt in med kampanjer som



anställda på Emerse ställt in. Vårt program föreslog inställningar som i medel gav en klickchans på 0,13%, medan Emerse anställdas inställningar i medel gav 0,12% klickchans.

Vår slutsats i detta projekt är att maskininlärning har stor potential när det kommer till marknadsföring på internet, men att det kommer krävas mer arbete för att en modell som vår ska kunna bli helt självgående. I nuläget fungerar den bättre som ett assistentverktyg för de som redan arbetar med att ställa in reklamkampanjer. Däremot tror vi att med internetreklambranschens konstanta tillväxt under de senaste åren så kommer denna teknik att bli mer och mer vanlig, samt bättre och bättre på att rikta rätt reklam till rätt personer.