

Capstone Final Report: Heatmap Anomaly Detection

Xinyi Chen (xc2719), Martin Fluder (mf3590), Jean Law (jl6642),
Ling Lin (ll3719), Yue Yin (yy3414)
Mentored by Jan Benzing and Matt Merenich
Criteo

September 20, 2024

Contents

1	Introduction	2
2	Broken Banner Detection in Online Advertisements	3
2.1	Datasets	4
2.2	Ground Truth Labels	6
2.3	Evaluation	7
2.4	Existing Work and Technology	8
3	Unsupervised Learning Methods	8
3.1	Data Preprocessing and PCA	8
3.2	Combined Heatmap Dataset and Performance Metric Dataset	9
3.3	Unsupervised Learning Models	9
3.4	Conclusions and Future Directions	12
4	Feature Extraction from Deep Learning Models	12
4.1	Pre-processing and Pre-Trained ViT and ResNet	13
4.2	Fine-Tuned ResNet Models	13
4.3	Results	14
4.4	Conclusions and Future Directions	15
5	Statistical Modelling: Likelihood Ratio Test	15
5.1	Methodology and Tests	16
5.2	Results	18
5.3	Future Work	19

1 Introduction

In today’s digital landscape, advertisements are everywhere, infiltrating our daily routines with remarkable frequency. From brief video clips that precede YouTube videos to the banners and full-page advertisements that we need to close to access desired webpages, online advertising is omnipresent. This prevalence is underscored by the fact that for technology titans such as Google, a substantial portion of revenue (approximately 80% as of 2021 [1]) is generated from online advertising, massively surpassing income from hardware sales and other products that seem more visible to the consumer. Similarly, but perhaps less surprisingly, 98% of Facebook’s revenue comes from advertisement sales [1].

This Capstone project is in collaboration with Criteo, a prominent French advertising company that reported a revenue of approximately 2 billion USD and showed 1.85 trillion advertisements during the financial year 2023, indicating its substantial presence in the advertising sector [2]. While smaller compared to industry giants like Google, whose advertising revenue exceeds 200 billion USD annually, Criteo exemplifies significant success and expertise in the highly competitive and dynamic field of digital marketing.

The core of modern advertising lies in the powerful use of data analytics and real-time decision-making processes. One of the most critical mechanisms in this sector is real-time bidding, where, upon a user’s visit to a webpage, an auction takes place among advertisers. They compete to display their ads, leveraging predictive algorithms to determine the value of presenting their advertisement to that specific user based on data, such as cookies and browsing history. These algorithms are essential for determining not only the likelihood of a user engaging with the ad but also calculating the bid for the advertising space. This is crucial for optimizing the click-through rate, which, in turn, maximizes the advertisement’s effectiveness and return on investment.

In this Capstone project, we will investigate a critical – yet often overlooked – aspect of digital advertising that significantly impacts a company’s revenue: the identification of broken banners in online advertisements. Advertisement banners – the usual banner type advertisements shown on many online webpages – can malfunction for various reasons. Issues such as lack of user permission, broken links to landing pages, faulty sizing of the banner for the ad space, or the rejection of third-party cookies by the user’s browser can render these advertisements non-functional.

Considering that companies like Criteo display approximately 1 billion advertisements, corresponding to 25 million banners, on a daily basis, the presence of broken banners signifies a significant loss in revenue. The early and accurate detection of these broken banners is crucial to mitigate financial impact. Therefore, this project will focus on developing efficient methodologies for identifying broken banners, ensuring the continuous effectiveness and revenue-generating potential of Criteo.

For this project, we investigated a variety of different methods and approaches to classify broken banners based on heatmaps created from click locations. This is a largely unexplored problem in the literature, and, thus, we built and tested all our methods from scratch. In the main body of the report, we shall focus on the following three instructive approaches that rely on substantially

different underlying ideas:

1. **Unsupervised Learning Methods:** These methods rely on classical unsupervised clustering methods and perform well given enough “training data”. They do not transfer well to different banner types, and, therefore, would require special training on each banner-type.
2. **Deep Learning Feature Extractors:** This approach leverages the feature vectors from pretrained and fine-tuned deep learning models. We extract five distinct feature vectors based on whose cosine similarity we perform zero- and one-shot classification. This method performs well and transfer to different banner types with little tuning, but its inference speed provides a significant performance bottleneck.
3. **Likelihood Ratio Test (LRT):** Our final model is based on extracting empirical likelihoods for the “existence” of a heatmap. We estimate these by assuming that the heatmap clicks were sampled from an underlying two-dimensional probability distribution. Using the likelihood ratio test, we then discern if a heatmap aligns more with a distribution characterizing unbroken or broken banners. This approach is lightweight, interpretable and performs well across banner types, making it our preferred model for real-world implementation.

In the remainder of this report, we first carefully set up the precise problem statement and analyze Criteo’s datasets. We then explore and describe the data and introduce our evaluation metrics. We then outline our three approaches of choice, based on unsupervised learning methods, deep learning, and the likelihood ratio test. We especially highlight the pros and cons of each method and provide suggestions for future refinement. We conclude with a brief discussion, future directions, and an estimate of the business impact of our best-performing model.

2 Broken Banner Detection in Online Advertisements

This Capstone project aims to develop an efficient, unsupervised, or self-supervised data processing pipeline and machine learning model to quickly and efficiently classify whether an advertisement banner is broken or not. Banners vary in sizes and shapes depended on the user’s device and browser such as phone, tablet, or computer. Furthermore, the banners can display a types and numbers of products (see for example Figure 1), which can be arranged in various ways; they can be aligned horizontally, vertically, in a grid-like structure, or mixed. For example, the top left image in Figure 1 is a mixed arrangement, while the bottom left corresponds to a horizontal alignment. As we shall see, the feature in our datasets that captures this type of product arrangement is called the `grid_id`.

We are interested in determining whether a banner is broken based on users’ behaviour. For example, we can try to determine whether a banner was broken based on users’ click locations.¹ A convenient way to visualize this “global” click behaviour is through click-heatmaps (see the corresponding heatmaps in Figure 1). In the following, we shall define a **banner type** to be a tuple consisting of a banner size (height and width) together with the number of products and

¹A refinement of this approach is to further include the timestamps of these clicks. We refer to Section 5.3 for some work in that direction.

a `grid_id` (arrangement-type of products). Such a banner type’s heatmaps will necessarily be the same independent of the actual products shown to the particular users. Therefore, heatmap banners are uniquely and systematically organized by this tuple. Furthermore, we shall call a **banner** an instance of a banner type on a particular domain. In other words, it corresponds to a choice of banner type together with a domain id. As we shall see, for most of our project we focus on banners of a specific size (250×300 pixels), and banner types together with their corresponding heatmaps are uniquely identified using their `grid_id` and the number of products.

The goal for this project is to promptly detect banner issues, leveraging users’ click behavior, even with minimal data. Key features of the project include:

1. Generalizability using *zero-shot learning* for unseen banner types.
2. Addressing class imbalance (less than 10% of banners are broken).
3. Early detection of broken banners, possibly with only around 200 clicks, utilizing metrics and data augmentation.
4. Lightweight and efficiently implementable for processing millions of banners daily, assuming the algorithm runs daily on an average server.

2.1 Datasets

Criteo has provided two types of datasets: the “heatmap” dataset and the “performance metric” dataset. The heatmap dataset contains users’ click behavior categorized by “`grid_id`” and “domain,” while the performance metric dataset is a breakdown of various metrics for the same banners. Here, domain corresponds to the (anonymized) root domain where the specific banner type was displayed. For each dataset, Criteo provided us with two banner types featured with different numbers of product, one with three products and another with six products. In the following, we shall refer to the three-product dataset as **data I**, and the six product one as **data II**. See Figure 1 for some examples from data I (left) and data II (right).

Heatmap Dataset

The heatmap dataset consists of an aggregation of users’ clicks collected by Criteo over a two-month period without any cleaning applied. Each click on a banner corresponds to a tuple in this dataset and is uniquely identified by the key (`domain`, `grid_id`, `click_x`, `click_y`), where `click_x`, `click_y` correspond to the coordinates of the clicked pixel.

More precisely, the dataset consists of the following attributes:

1. Domain: represents the root domain on which a banner type was displayed.
2. Grid.id: specifies the grid layout and product arrangement of a given banner.
3. Click.x: x-coordinate of the pixel being clicked on.
4. Click.y: y-coordinate of the pixel being clicked on.

5. Display_height, display_width: banner pixel dimensions, fixed to 250×300 in data I and data II.
6. Clicks: number of clicks recorded for the specific pixel.

Data I In the three-product heatmap dataset (data I), a total of 26M clicks are provided. Each of the clicks belongs to one of two grid_ids, 333519 and 333346, representing different arrangements of products within a banner. Each grid_id is associated with approximately 800 domains. Since the number of products and the dimensions of the banners are fixed in this dataset, the two grid_ids define two separate banner types, which display distinct characteristic click behaviors. In Figure 1, all clicks associated to a given grid_id are aggregated into a single click heatmap, and, therefore, represent a global view over the user behaviour for this banner type. We observe that the grid_id 333519 represents a mixed arrangement, while grid_id 333346 is of horizontal type. We notice an aggregation of clicks on the bottom left, corresponding to the “X” button to click out of the advertisement. The bottom right corner is devoid of clicks, which is something we attest to an overlapping/non-clickable area.

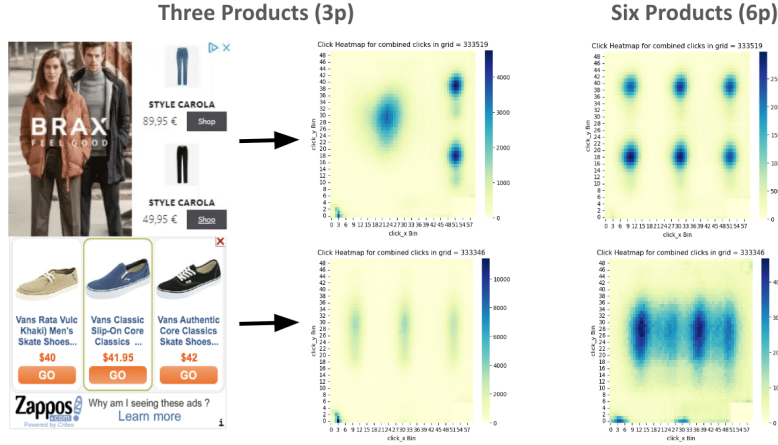


Figure 1: **Left:** Two three-product examples of advertisements by Criteo, which could lead to the user behaviour that generates the three-product heatmaps for the banner types of data I. **Right:** The heatmaps for the two banner types for the six-product dataset (data II).

Data II For the six-product heatmap dataset (data II), Criteo provided the data in two CSV files. The dataset is mainly used to assess transfer learning; *i.e.* whether a model trained/tuned on data I transfers well to banners with a different number of products, while retaining the overall size of the banners. Depicted on the right side of Figure 1 shows the aggregated clicks on six products heatmap dataset for the two banner types. The grid_id 333519 and grid_id 333346 are again of mixed and horizontal type, respectively.

Performance Metrics Dataset

The performance metrics dataset consists of metrics data for a superset of domains in the heatmap dataset for the same two grid_ids. The Criteo team acquired this data over roughly one month, which is a subset of the two months over which they acquired the data for the heatmap dataset. We have been provided the raw (anonymized) data without any cleaning applied. In addition to domain, grid_id, number of clicks and displays, the performance metrics dataset contains the following features, which we shall use in the following:

1. Landed_clicks: number of clicks that successfully led to a landing page.
2. Closing_events: number of times users closed the ad.
3. Avg_last_second_framerate: average framerate in the last second.
4. Sov_short_ttc_global: difference between the clicks and display timestamps across all domains.
5. Sov_short_ttc_score: $B - A$, where we defined A = percentage of clicks within one second for *global distribution*, and B = percentage of clicks within one second for a *domain*.

2.2 Ground Truth Labels

To assess our models, we first determine the “ground truth,” by identifying the broken banners in our datasets. To do so, we defined a “clearly broken banner set” as the set of banners which are broken under the following steps:

1. First, we bin the clicks into 5×5 pixel boxes (which reduces the pixel size from 250×300 to 50×60 pixels).
2. Then, we use noisy bootstrap enhancement (*i.e.* bootstrap the location of a click per banner and perturb it with Gaussian noise of fixed standard deviation) to enhance each heatmap to 5000 clicks.
3. If the determining structure (*e.g.*, triangle or line-structure for data I) of clusters for the grid_ids, is not clearly identifiable even with this noisy bootstrap enhancement, we define the banner as broken.
4. We do not care about the rest of the heatmap if this pattern is clearly defined.

Conditions 3 and 4 imply that we identify a banner as “clearly broken” if users primarily click on the “X” to close the advertisement. While the banners might not inherently be broken, from a business perspective, they are considered anomalous since they are ineffective at engaging users with the products. Figure 2 shows some representative examples of clearly broken banners, contrasting them with the heatmaps shown in Figure 1, which display the desired triangle and vertically aligned structure, respectively.

We have used various techniques to narrow down the search space and identify the clearly broken banners, but effectively, it was an effort primarily done manually. In data I, we identified 75 clearly

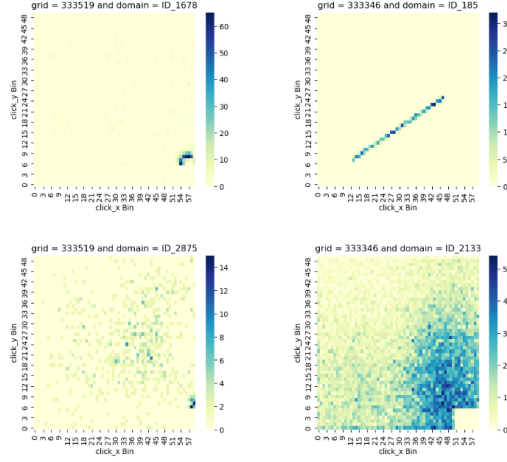


Figure 2: Examples of heatmaps for broken banners. None of them display the characteristic structure of the underlying banner types.

broken domains out of 872 for grid_id 333519, roughly 8.6%, which align with Criteo’s estimate of 5-10%. For the grid_id 333346, however, we found 137 broken banners out of 861, corresponding to 15.9%, which is significantly more than expected. The reason for this remains unclear to us and our mentors.² In data II, we identified 86 clearly broken domains out of 2061 for grid_id 333519, roughly 4.2%. For the grid_id 333346, we found 153 broken banners out of 969, corresponding to 15.8%.

2.3 Evaluation

In our evaluation process, we rely on the “macro F1-score” as our primary metric for assessing the performance of our models. The macro F1-score is calculated using the following formula

$$\text{macro F1} = \frac{1}{N} \sum_{i=1}^N \text{F1}_i,$$

where N is the number of classes and F1_i is the F1-score for class i . The F1-score for each class is computed as the harmonic mean of precision (P_i) and recall (R_i)

$$\text{F1}_i = \frac{2 \times P_i \times R_i}{P_i + R_i}.$$

In the following, whenever we refer to the F1-score, we implicitly mean the macro F1-score unless stated otherwise.

Precision represents the ratio of true positive predictions to the total number of positive predictions for a specific class, while recall represents the ratio of true positive predictions to the total

²Somewhat speculatively, it could be due to the fact that this banner-size is exclusively displayed on Android, which is anecdotally more error-prone.

number of actual positive instances for that class. The macro F1-score then combines these values to provide an overall evaluation of the model’s performance across all classes. This approach ensures that each class receives equal consideration, regardless of its frequency in the dataset.

Compared to other metrics like accuracy or micro-averaged F1-score, the macro F1-score offers a more balanced evaluation, especially when dealing with our imbalanced datasets. It takes into account both false positives and false negatives, providing a comprehensive understanding of the model’s ability to classify data from all classes. By using the macro F1-score as our primary evaluation metric, we ensure a thorough assessment of our models’ performance in detecting broken banners.

2.4 Existing Work and Technology

In the literature, there does not exist publicly available resources dealing with the problem to identify broken advertisement banners. Many advertisement companies will have their own proprietary methods in place, but sharing them would only help their competitors. Consequently, we dedicated considerable effort to devising our own solution path. Our inspiration comes from various cutting-edge techniques in heatmap analysis, computer vision, and anomaly detection, which we amalgamate into a comprehensive approach tailored to our problem.

Drawing insights from diverse fields of research informs our methodology, as we strive to integrate state-of-the-art techniques to address the unique challenges associated with broken banner detection. In contrast, Criteo’s existing system for identifying anomalous and broken banners relies solely on explicit thresholds derived from a banner’s performance metrics. An advancement in their system would involve incorporating techniques that handle heatmap analysis, potentially extending to scenarios with a limited number of clicks.

3 Unsupervised Learning Methods

In this section, we introduce our implementation of unsupervised learning models. These methods show strong performance but require a significant amount of data per banner type to reliably identify broken banners. Our approach involves using an overall data preprocessing pipeline that includes principal component analysis (PCA) to reduce the dimensionality of the heatmap data vectors and combining the two provided datasets. Following this, we tune the hyperparameters (decision thresholds) on various unsupervised learning algorithms. Finally, we combine the results of these models to obtain a final performance metric.

3.1 Data Preprocessing and PCA

Our heatmap datasets contain pixel-level clicks for each grid_id and domain combinations. To reduce the dimensionality of our vectors, we first bin the clicks as described in section 2.2, where we bin each click into the respective 5×5 pixel box. This reduces the heatmap vector from 250×300 to 50×60 dimensions. This binning significantly reduces the dimensionality of our heatmap vectors

but still contains a lot of spurious and sparse directions. Therefore, it is reasonable to apply further basic dimensionality reduction techniques, such as PCA.

We first normalize each heatmap vector h_i to have an equal number of clicks,

$$h_i \rightarrow \tilde{h}_i = \frac{h_i}{\text{total number of clicks in } i^{\text{th}} \text{ heatmap}}$$

before applying the PCA transformation to the matrix $H = \{\tilde{h}_i\}_{i=1}^N$.³ To streamline the pre-processing of our heatmap dataset, we create a pipeline using the *pca_pipeline.py* script. This script accepts parameters like *grid_id*, *n_components* (the desired number of PCA components), and *data_directory* (the location of the preprocessed data).

3.2 Combined Heatmap Dataset and Performance Metric Dataset

We also seek to investigate whether combining the click-based data from the heatmap dataset together with the performance metrics dataset can lead to improved clustering results. We combine the two datasets as follows:

1. PCA on the heatmap dataset down to 200 components.⁴
2. Normalize those PCA vectors together with the five (non-collinear) performance metric dataset features (*landed_clicks*, *closing_events*, *avg_last_second_framerate*, *sov_short_ttc_global*, and *sov_short_ttc_score*).
3. Run PCA on these 205 feature vectors.

This approach yields a combined set of PCA vectors for both datasets, which we shall denote by PCA. Figure 3 shows the first and second PCA directions. The unbroken banners seem clustered in the center, while the broken banners are clearly "noise" away from these central clusters. While the PCA-pictures look promising, running the same type of clustering hyperparameter tuning approach as for the heatmap-only dataset (see Section 3.3) leads to (significantly) worse-performing models. The inclusion of the performance metric dataset seems to significantly affect the variance of the PCA-vectors and therefore pick directions that are "worse" at distinguishing noise versus actual useful features.

3.3 Unsupervised Learning Models

We now delve into the application of various machine learning approaches for broken banner detection and focus on six distinct methodologies to serve as baseline models: K-Means clustering [3], Isolation Forest [4, 5], one-class SVM [6], KNN [7], and DBSCAN [8, 9, 10]. The application

³Not normalizing the vectors by click numbers, results in much less expressive PCA components; in particular, the PCA vectors pick up the number of clicks as a feature, which is not something we are interested in capturing.

⁴The choice 200 components is somewhat arbitrary, but the actual number does not affect the end result by much. While fewer components, for example 20 components, seems to perform slightly better, the overall results are still inferior to the heatmap-only clustering.

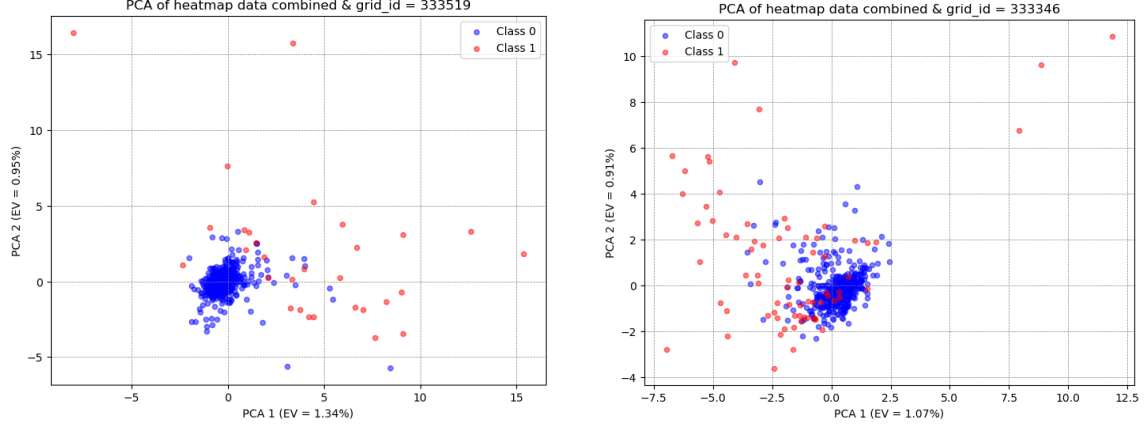


Figure 3: Plot of the first and second $\widetilde{\text{PCA}}$ directions from the combined (PCA-ed) heatmap and performance metric dataset.

of these algorithms in various domains, from network security to environmental monitoring, shows their versatility and effectiveness in detecting deviations from established patterns [11]. Each of these techniques is quite well-known for its capability to identify outliers without the need for labeled data.

In this phase, we utilize the three-product combined heatmap dataset (data I) for training and tuning purposes and the six-product combined heatmap dataset (data II) for testing. To ensure an unbiased evaluation of the selected methods, we first conduct hyperparameter tuning on data I to optimize the performance of each model. Subsequently, these optimized models are applied to data II. This includes the PCA vectors and the choice of PCA dimension. Our tuned hyperparameters and the results of the best performing models on both data I and data II can be found in Table 1.

Let us briefly summarize our findings:

- **K-Means:** this is an unsupervised machine learning algorithm, categorizes datasets into K groups based on the proximity to their means [12]. In the process of optimizing our model, we focused on tuning hyperparameters. Our findings indicated that adjusting the number of clusters (K) minimally impacted the F1-score, leading us to maintain K=2 for its simplicity.
- **Isolation Forest:** this algorithm excels in anomaly detection within high-dimensional datasets and does not depend on distance or density measures to identify anomalies, which is useful for our project, as anomalies in banner heatmaps are typically rare and significantly different from normal data points. We focused on tuning two main hyperparameters: the number of PCA dimensions and the number of estimators in the Isolation Forest.
- **One-Class SVM:** our choice of OCSVM was motivated by its robustness in handling unsupervised learning tasks, particularly in distinguishing outliers from normal data points in a transformed feature space. This method is also famous in solving for high-dimensional data, since OCSVM focuses on isolating normal observations from anomalies.

Method	Hyperparameters	Train (data I)	Test (data II)
1. K-Means	PCA dimension = 2 $k = 2$	0.38	0.45
2. Isolation Forest	PCA dimension = 10 n estimator = 100	0.91	0.77
3. One-Class SVM	PCA dimension = 178	0.46	0.26
4. KNN	PCA dimension = 10 $k = 10$ threshold = 90%	0.84	0.85
5. DBSCAN	PCA dimension = 8 epsilon = 5 MinSample = 10	0.90	0.77
6. Combined	All 5 models combined	0.94	0.79

Table 1: Summary table of model performance using several types of algorithms on data I and data II.

- **KNN**: this algorithm serves as a foundational approach and baseline model to handle the classification and unsupervised anomaly detection tasks. Since it is a non-parametric approach, it does not need to make underlying assumptions about the distribution of the data, and it relies on the similarity of feature vectors to classify new data points and predict outcomes. To optimize the model performance, we conducted hyperparameter tuning by adjusting parameters such as the number of nearest neighbors, threshold, and PCA dimension.
- **DBSCAN**: this is a density-based clustering algorithm [13]. Epsilon determines the radius within which points are considered neighbors, and minimum samples specifies the minimum number of points required to form a dense region. It groups together data points that are closely packed, while marking points that are in low-density regions as outliers or noise.
- **Combined Models**: For each data point, we aggregated labels predicted by a randomly selected subset of the models mentioned earlier. This aggregation involved checking if the count of '1's in the collective output of the selected models surpassed half the total number of models in the ensemble. If so, the aggregated prediction was set to '1'; otherwise, it was set to '0'. Among all possible combinations, amounting to 2^5 , we discovered that the strategy of combining all models yielded the highest F1-scores.

3.4 Conclusions and Future Directions

In conclusion, our exploration of unsupervised learning/clustering methods for identifying broken heatmaps provides valuable insights into the characteristics of the dataset as well as the performance using a significant amount of data per banner type. By employing techniques such as PCA to simplify the data and merging different datasets, we have taken important steps towards improving the base clustering algorithms. Despite facing challenges like data variability and anomaly detection, algorithms like Isolation Forest and KNN demonstrate robust performance. Moreover, our ensemble approach, which combines the strengths of multiple models and generally adds significant robustness to models, shows promising enhancements in performance, highlighting the effectiveness of such methods.

It is important to note that this method does not generalize well across different banner types, as evidenced in Table 1, and requires a significant amount of data to cluster on for each specific banner type.⁵

Moving forward, future research could concentrate on refining techniques for tuning hyperparameters to boost model performance across various datasets and domains. Additionally, delving into advanced anomaly detection algorithms and ensemble learning techniques could provide valuable insights into addressing the complexities of detecting broken heatmaps. However, in the following we shall leverage deep learning techniques to develop more sophisticated methods, which intuitively should be more adaptable from banner type to banner type.

4 Feature Extraction from Deep Learning Models

Until now, our focus has been on anomaly detection methods that depend on substantial data per banner type. For instance, to classify a heatmap vector as noise, it is necessary to have a cluster of corresponding non-noise vectors (domains). Furthermore, in the previous section we saw that the unsupervised learning methods did not perform well transferring to different banner types. Therefore, to improve their performance, we would have to run a separate training algorithm for each banner type. It is technically feasible but would require the Criteo team to “train” a new model for each new banner, which is resource intensive. It is, therefore, essential to explore global methods that judiciously identify broken banners (almost) independent of the underlying banner type.

The remainder of this report will focus on “one-shot” and “zero-shot” methods. Here, we define **one-shot** models as those that require only a single representative heatmap to classify other heatmaps of the same banner type. Similarly, **zero-shot** methods require no additional data and are supposed to be transferable across banner types.

In the current section, we employ computer vision deep learning models as feature vector extractors. These features encode the underlying structure of a visual representation of the banner heatmaps, and we can assess whether two banners are related using cosine similarity. We generate

⁵Although the F1-score of 0.79 is not much lower than some of the later models, it is mainly due to the heatmaps in data I and data II being fairly similar. For instance, switching to different-sized banner types will notably lower this score.

five distinct feature vectors; two from pretrained vision models, ViT [14] and ResNet-50 [15], and three from fine-tuning ResNet-50 on different synthetic tasks and data.

4.1 Pre-processing and Pre-Trained ViT and ResNet

In order to use pre-trained models, such as the ViT and ResNet-50, we preprocess the heatmaps into images. To accentuate the structure of the heatmaps, we (1) bin the heatmaps into 5-by-5 boxes, (2) perform a noisy upsampling to 50'000 clicks (*i.e.* we bootstrap sample clicks and add Gaussian noise to their location), and (3) rescale the number of clicks to match standard RGB-encoding (we stack the three greyscale channels). See Figure 4 for three such examples. Finally, we re-scale the images to match the input dimensions, $3 \times 224 \times 224$, of the models.

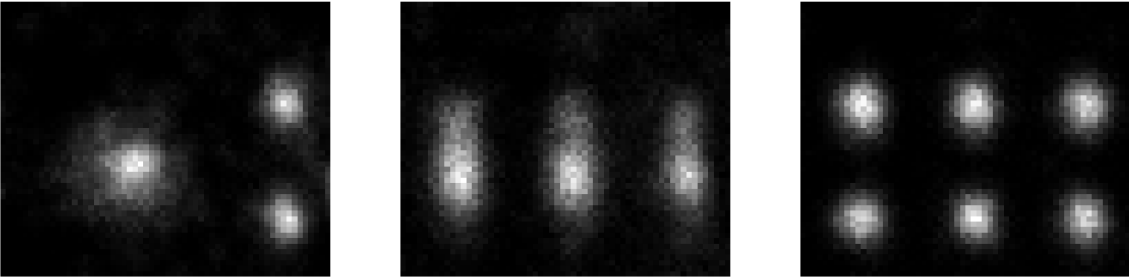


Figure 4: Three examples of heatmaps upon preprocessing into RGB-images.

Upon preprocessing, we can use pretrained ResNet-50 or ViT models from Hugging Face as feature extractors, which we extract as the vectors prior to the classification layer. This results in 768- and 2048-dimensional feature vectors for the ViT, ResNet, respectively. These models are very good at identifying patterns and distinguishing different image-contents, and we imagine that – like humans – they can distinguish broken from non-broken banners. Applying Maximum Variance Unfolding (MVU) [16] to these vectors indeed shows significant clustering of broken and unbroken banners and lends credibility to this assertion.

4.2 Fine-Tuned ResNet Models

We can improve these pre-trained models further by fine-tuning them. We specifically fine-tune the ResNet-50 model on three different datasets of synthetic heatmaps and tasks. This allows us to generate three additional feature vectors.

The first fine-tuning task consists of training the model on detecting the number of Gaussian clusters in synthetic heatmaps. We randomly generate a number (between 0 and 20) of Gaussian clusters with random centers and covariance matrices on the 224×224 grid and overlay each of these pictures with noise drawn from the uniform distribution on the two-dimensional image. We further add a 21-fold softmax classification head to the model and train the model for 50 epochs,

where in each epoch we generate a novel training set of 1000 synthetic heatmaps. We reach a test accuracy of over 80% and extract the corresponding 2048-dimensional feature vectors.

The second fine-tuning task consists of training the model to detect the *center* of a random number (between 0 and 6) of clusters. We develop two distinct ways of generating training data; the first being much less structured (we can have overlapping and much smaller-sized clusters), while the second includes conditions to avoid overlapping clusters and contains larger and more defined clusters as well as additional noise. For both these synthetic datasets, we train the model on minimizing the “Chamfer” distance, which specifically aims to reduce the distance between (unordered) sets of values

$$d_{\text{CD}}(\{X_i\}_i, \{Y_j\}_j) = \sum_i \min_j \|X_i - Y_j\|_2^2 + \sum_j \min_i \|X_i - Y_j\|_2^2, \quad (1)$$

where $\{X_i\}_i$ and $\{Y_j\}_j$ are the two sets of values – cluster locations in our case – we are comparing. We run these models on around 100 epochs of 1000 synthetic images each and extract the corresponding 2048-dimensional feature vectors.

4.3 Results

To create a predictive model from these five feature vectors, we compute their cosine similarity between a sample heatmap and an “anker” heatmap. The resulting performance is significantly dependent on the choice of the anker heatmap. We employ two strategies:

- **One-shot:** For one-shot learning, we pick a “representative” heatmap as the reference against which we measure cosine similarity for each banner type. The representative choice of banner is determined by picking the domain for which we have the most clicks among domains with landing rate above 80%.⁶
- **Zero-shot:** For the zero-shot approach, we pick as anker an aggregate of banners which we classified as broken. Specifically, for data II, we pick an aggregate of the broken banners of data I and vice-versa in order to avoid data leakage.

For both the zero-shot and one-shot approaches, we determine the thresholds for each feature vector individually and then use majority voting to select the best performing set in a given training set. Our best results for the transfer learning are summarized in Table 2.

Method	Train	Test	Thresholds (RN, loc1, loc2, count, ViT)	Test (F1-score)
zero-shot	data I	data II	(0.70, 0.78, 0.51, 0.81, –)	0.78
one-shot	data II	data I	(–, 0.80, –, 0.86, 0.72)	0.86

Table 2: Summary table of model performance for the deep learning models. The thresholds are given for the cosine similarity threshold based on the pre-trained ResNet, fine-tuned Resnet on location I, II, count and pre-trained ViT, in that order. By a dash we indicate that this cosine similarity did not feature in the majority voting based on the training data.

⁶The same logic of determining the representative banner is applied in the one-shot approach of Section 5.

Both cases demonstrate robust performance when transferring between data I and data II, even though the corresponding heatmaps have quite different structures. For both zero- and one-shot approaches, one of the pretrained feature vector can be neglected (ViT for zero-shot and ResNet for one-shot). We believe that both are actually quite comparable for our task, since they are pretrained on relatively similar tasks, which consist of distinguishing cats from dogs (rather than heatmaps). For the one-shot model, we can additionally neglect the fine-tuned model on the second location task.

4.4 Conclusions and Future Directions

The deep learning approach performs quite well in terms of the (macro) F1-score, and the underlying methods and ideas are intuitive and seem quite solid. However, a big draw-back is that inference, which consists of preprocessing with upsampling and model inference, is relatively slow. Given the amount of data Criteo deals with on a daily basis ($\sim 25M$ banners), running such a model would require a significant amount of hardware and time to be feasible in practice. We therefore developed lighter-weight models in the following, such as the ones in Section 5.

From an academic standpoint, there are a variety of potential future improvements one could consider for this approach:

- The (synthetic) data generation is imperative in the models’ performance. It would be interesting to match the data closer to the actual underlying heatmaps and fine-tune our models on that.
- Additionally, there are further synthetic tasks, which could boost performance. For example, training on predicting the size as well as the location of the clusters or distinguishing between “symmetric arrangements of clusters” and “no structure/noise,” *etc.*⁷
- Finally, we briefly explored a contrastive learning approach, where the idea is to train an encoder-decoder model that recreates a heatmap while simultaneously distinguishing the input from a heatmap of different banner type. This might lead to more faithful representative feature vectors in a compressed vector space.

5 Statistical Modelling: Likelihood Ratio Test

Even though the state-of-the-art computer vision models in the previous section show immense potential in resolving the zero- and one-shot problems, they cannot be easily deployed on Criteo’s servers due to the required high computational power.

We hence start looking into more computationally efficient methods that can classify whether a banner is broken. A notable and powerful method is based on the likelihood ratio test (LRT).⁸ For this test, we consider each click on a given heatmap to be an independent draw from a two-dimensional statistical distribution over the banner. Thus, we can assign an empirical probability

⁷We have implemented a naive version of the latter, but it did not perform well.

⁸We thank Prof. Chen for suggesting this idea and for taking the time to discuss it with us on several occasions.

for each click given the underlying distribution. Intuitively, if we choose the underlying probability distribution to be representative of an unbroken banner allows us to compute the cumulative probability of a heatmap. If this probability crosses a threshold and becomes too small (compared to some baseline), we might be dealing with a broken banner. The objective then is to determine a clean and meaningful underlying statistical distribution for a given banner type. Given such a distribution, we can compute the likelihood for the “existence” of a certain heatmap.

In the LRT, we have two competing hypotheses for the underlying distribution,

- (1) the underlying distribution represents a banner that is not broken, that we call a “good” banner (H_0), or
- (2) it represents a broken, or a “bad” banner (H_1).

Then, for each sample banner the LRT returns a ratio that indicates whether this banner resembles a good or bad banner. A visual representation of this classification process can be found in Figure 5. Mathematically, we formulate the hypotheses as follows:

H_0 : the banner is good: likelihood = $P[\text{clicks}|\text{banner good}]$

H_1 : the banner is broken: likelihood = $P[\text{clicks}|\text{banner broken}]$

LRT: log-likelihood ratio is given by the following expression:

$$\text{LRT} = \log P[\text{clicks}|\text{banner good}] - \log P[\text{clicks}|\text{banner broken}] \quad (2)$$

Thus, the LRT method essentially boils down to clever and meaningful choices in representative “good” and “bad” distributions over the given banner dimensions. It is noteworthy that upon normalization, the LRT method is essentially similar to computing (cosine) similarity between heatmap vectors and the representative good/bad banners and subsequently take the ratios of these similarity values.⁹

5.1 Methodology and Tests

As discussed, the choice of the underlying distribution (*i.e.* the representative banners) for the working (called “good” banner from this point onwards) and broken (“bad”) banners is crucial. One way of choosing the underlying distribution would be to take the “best-looking” heatmap from the dataset. This would lead to a one-shot learning approach, as we require a representative sample for each banner type. Alternatively, we can generate zero-shot approaches by generating our own representative distributions.

Hence, our methodology mainly consists of finding underlying distributions for both the good and bad banners that would allow us to best distinguish between the two categories. This approach consisted of two phases.

1. **Test 1:** (one-shot learning methods) We did three separate trials:

⁹The main difference lies in taking log-probabilities (in the LRT case) rather than actual probabilities.

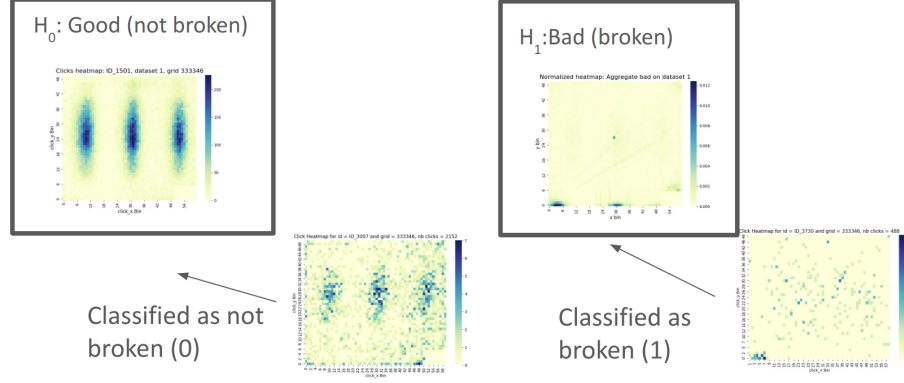


Figure 5: Example of how two specific banners would be classified by the LRT method using two underlying distributions. The heatmap on the bottom left has a log-LRT value that is “closer” to the non-broken representative, while the one on the bottom right is “closer” to the broken representative banner.

- (a) **Test 1a:** For the good banner, pick the banner that had the “best” contour; we filter the dataset based on *landed_clicks* > 0.8 and among those choose the banner for which we have the most number of clicks. For the bad banner, we create a banner by aggregating all the bad data from the other dataset, *i.e.*, for the bad banner for dataset I, II, we aggregate the clicks from all the broken banners from dataset II, I, respectively.
This approach produced the best results out of all one-shot learning approaches, with a macro F1-score of 0.95 and an area under the ROC curve (AUC) of 0.99.
 - (b) **Test 1b:** Next, we tested whether using a uniform distribution to represent the bad banners would improve the classification performance (using the same good banner). The macro F1-score dropped to 0.94 with an AUC of 0.98.
This shows that even across datasets, the broken banners resemble each other.
 - (c) **Test 1c:** As an attempt to improve the results for the one-shot learning from Tests 1a and 1b, we tested whether taking the sum (and normalizing) the two bad representative banner distributions used would yield better performance (*i.e.* for each of the 3000 bins on the heatmap, we sum the probability that this bin is clicked in the uniform distribution and that of the aggregate bad data from the other dataset and divide this sum by two to produce a new bad distribution) would yield better performance. The good banner remains unchanged.
This led to the same performance as in Test 1a with a F1-score of 0.95 and an AUC of 0.99.
2. **Test 2:** (zero-shot learning methods) Next, we looked into zero-shot learning methods, *i.e.*, avoid picking any banner from the same dataset to be the representative banner. We did three separate trials for the representative banner:
 - (a) **Test 2a:** In this case, the good banner is represented by a uniform distribution, and the bad banner is represented by aggregating bad data from grid_id 333346 in the other dataset, similar to Test 1. The best F1-score is 0.81 with an AUC of 0.76.

- (b) **Test 2b:** Similar to test 2a, the good banner is represented by a uniform distribution, and the only modification is that the bad banner is represented by aggregating all bad data from the other dataset, independent of `grid_id`. The best F1-score is 0.86 with an AUC of 0.77.
- (c) **Test 2c:** Similar to test 2b, but the good banner is represented by a “step” uniform distribution in which the probability density function is constant for $x \in [a, x_nr_bins - a]$ and $y \in [a, y_nr_bins - a]$, and 0, otherwise, where we chose $a = 5$ in our tests (see bottom left of Figure 6).

This has yielded the highest F1-score of 0.91 and AUC of 0.93.

For each of the above tests, we generate a classification for each banner using the following protocol:

1. Calculate the LRT ratio using the formula given by “**LRT**” in equation 2.
2. Normalize the output such that the ratio is a float between 0 and 1, 0 if the banner is closer to the representative good banner and 1 if the banner is closer to the representative bad banner¹⁰.
3. Use `np.linspace` to generate 100 thresholds (floats equally spaced between 0.01 and 1). For each threshold,
 - Calculate the prediction of the class of each banner: 1 if the normalized output is greater than the threshold, 0 otherwise.
 - Calculate the F1-score between the prediction and the labels.
 - If the threshold produces the highest-seen F1-score, save the F1-score and the threshold.
4. The classification for the banners is given by the classification produced by the threshold which yielded the best F1-score.

5.2 Results

After testing several different combinations of representative heatmaps, we found that the models from Tests 1a and 2c produce the best results.¹¹ We also tested the ability of our method to generalize to different banner types. We present the results from the best zero-shot learning and one-shot learning test in Figure 6:

For the case of Test 1a, we note that the broken banners from one dataset with a certain number of products transfer well to the ones with different numbers of products, provided the size of the underlying banner remains the same (250×300 pixels). However, upon changing the size of the banners away from 250×300 , the method does not transform as well. One reason for this is because the structure of the broken banners can be significantly different from size to size. For example, for certain banners we found that the “X”-button is in a different locations, which affects

¹⁰This is not strictly necessary but provides a way to easily compare thresholds across tests.

¹¹The source code for these two tests can be found on GitHub.

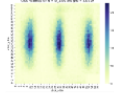


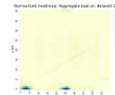
"Good" banner	"Bad" banner	Learning	Macro f1 (same dataset)	Macro f1 (different dataset)
		One shot	0.95	0.92
		Zero shot	0.91	0.91

Figure 6: Best zero-shot and one-shot LRT results from all of our tests

the representative power of an aggregate of broken banners. Additionally, we find that the choice of the representative good banner is crucial for good performance of the method. In particular, banners with only few numbers of clicks negatively affect the performance.

For Test 2c, a majority of bad banners have a disproportionate number of clicks on the frame of the heatmap. Conversely, good banners mostly contain clicks in the center of the heatmap. Therefore, while this method efficiently identifies broken banners characterized by excessive frame-clicks, it falls short in detecting banners with central clicks that deviate from the expected product/cluster-type structure. In our experience, the latter are generally much harder to identify using our techniques and require more fine-tuned models.

Overall, the key advantages of this method are that it is relatively straightforward to generalize to other banner types and is extremely fast with a total processing time of 0.05s/banner when we performed the computations on a local computer which used an Intel i7 processor, 4.7GHz with 16GB RAM.

5.3 Future Work

In follow-up work, we investigate whether we can detect explicit deterioration of banners over time. The intuition is that, given timestamped click data, we should be able to detect when a banner crosses an LRT threshold and deteriorates. In particular, we expect that once a banner breaks, a majority of clicks will be below the threshold and thus should provide a refined method of identifying broken banners.¹² Criteo provided us with a third dataset (data III) which contains banners with timestamped clicks. The data only covered roughly 12 days, and we were not able to find many examples of banners that seemingly broke during that time; *i.e.* most were either not broken at all or were already broken by the time of the first click in the dataset.

¹²In a batch-wise (rather than streaming) processing approach, one might encounter an overlap of clicks that happened when the banner was still intact and ones that arose after the banner broke. Such an overlap significantly affects the performance of the LRT. This is confirmed in our experiments, where we explicitly found that the rolling-window approach generally drops below the threshold earlier than the cumulative one.

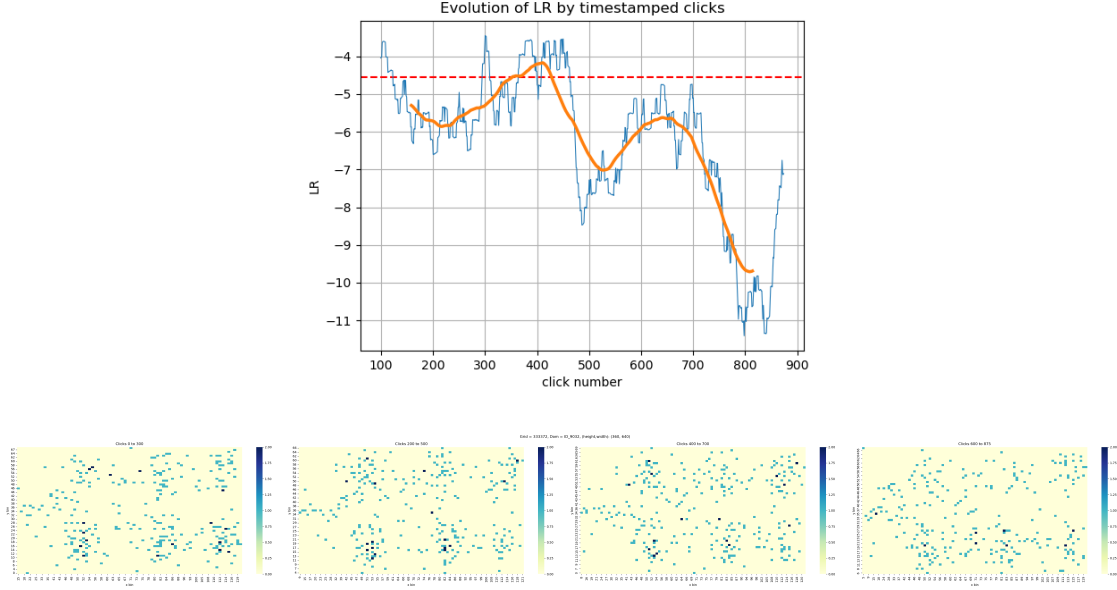


Figure 7: **Top:** Evolution of the (unscaled) LRT across time (ordered clicks); the LRT is computed in a rolling window fashion by aggregating the log-probabilities over 100 clicks. The dashed line represents the threshold, below which a banner will be classified as broken, and the orange line is a smoothed-out version of the LRT. **Bottom:** The heatmaps of the same banner across time. Each image corresponds to 300 clicks and contains the ordered clicks in the ranges 0 to 300, 200 to 500, 400 to 700, and 600 to 875. We observe that the initial structure of 8 products (4 on top and 4 on the bottom; the two leftmost clusters are much less pronounced in all heatmaps of this banner type) is essentially lost in the final picture.

In Figure 7, we show an example, where we can relatively clearly detect some deterioration. However, since such examples are rare, we leave further exploration for the future. In that context, it would be interesting to also explore a version of Shewhart individuals control chart, which allows for the detection of outliers from the “usual” behaviour.¹³

Furthermore, it would be interesting to study the LRT method using several types of “good” and “bad” representative banners, and thus create a variety of feature vectors (similar to the approach in Section 4). For example, we could imagine categorizing different kinds of broken banners and compute for each one of them a LRT feature, which we could then combine in some majority voting-type logic. One of these broken-banner category could consist of cases with disproportionate amounts of clicks on the frame of the banner, another one of cases with uniform distribution in the center (rather than distinct clusters), *etc.*

Finally, our base LRT approach provides a single feature for which we choose an explicit strict threshold. However, a more refined approach would allow for a “grey-zone” between broken and

¹³We thank Prof. Chen for suggesting to look into control charts.

unbroken banners, and if a banner falls within that grey-zone, further metrics could be used to determine whether it is actually broken. For example, one could look at the landing rate or other features of the metrics dataset as a deciding factor. Such methods could result in performance improvements, while retaining the speed and interpretability of the base LRT approach.

6 Conclusions and Business Impact

In this Capstone project, we explored the world of anomaly detection in the setting of advertisement banners. Our goal was to develop novel methods for detecting anomalous user behaviour based on their “average” click behaviour. To do so, we especially studied the impact of broken advertisement banners on click-location heatmaps, and how one can go about detecting them.

Throughout the project, we pursued several directions, starting from general dimensionality reduction and clustering methods, which performed surprisingly well (especially upon employing majority voting), but required a significant amount of training data for each banner type and are not robust upon transferring to different banner types. Since Criteo deals with a large amount of banner types, this would require a separate training for each one of them, which is not feasible in practice.

Therefore, we moved on to more sophisticated deep learning methods, where we extract features of pretrained and fine-tuned ViT and ResNet-50 models. In particular, to improve the power of their features, we fine-tuned ResNet-50 on a variety of synthetic tasks and datasets. By doing so, we end up with five different feature vectors. We then compute cosine similarity for these feature vectors and thus classify banners by thresholding over the resulting similarity scores. These models show a lot of promise in terms of transferability between banner types. We believe further training on clever synthetic tasks and datasets could significantly improve such models. However, the models are intrinsically slow and notoriously require powerful hardware. Given the amount of data Criteo handles on a day-to-day basis, we deemed it impractical to implement such models.

Finally, we investigated a statistics (likelihood-ratio) inspired technique, where we measure the “similarity” of a heatmap and an underlying representative by computing the heatmap’s probability for arising from the representative’s distribution. This method boils down to clever choices of underlying distributions. It is extremely lightweight, easy to understand and interpret and in our experiments performs extremely well as zero- and one-shot learning methods. We further discussed several refinements and interesting directions for the future work.

Given the speed and hardware constraints, we propose our best LRT zero- or one-shot methods, described in Tests 1a and 2c in Section 5 (see also Figure 6), for real-life implementation. The one-shot method performs better but is reliant on a “clean” representative heatmap for each banner type, while the zero-shot method is supposed to be plug-and-play for any banner type.¹⁴ Given the performance of these models, we can calculate their business impact for Criteo. The top-performing model achieves a recall of approximately 0.9. Criteo’s team suggested that they are dealing with roughly 25M banners per day. Given the number of broken banners in our datasets as well as Criteo’s own estimation, we assume that approximately 8% are broken, which leaves ~ 2 M broken

¹⁴As mentioned in Section 5, there might be some caveats when transferring to drastically differently sized banner types.

banners per day. Furthermore, from the performance metrics data we have received, we can infer that each banner gets approximately 1 click on average per day. Assuming a very conservative cost per click (CPC) rate of \$0.1 and Criteo earning a 20% commission, the revenue from banners correctly identified by our algorithms amounts to approximately \$13 million annually.¹⁵ When compared to Criteo’s reported revenue of \$1.95 billion in FY 2023, it is evident that our models contribute modestly to the overall revenue growth.

Acknowledgements

We would like to thank Jan Benzing, Matt Merenich and Criteo for their guidance on this project, providing us with an interesting problem and insights into the advertisement business. We would also like to thank Professor Sining Chen for her valuable suggestions and guidance in this project.

References

- [1] S. Ovide, “Google and facebook’s ad empires,” 2021. Available at nytimes.com.
- [2] C. S.A., “Criteo reports fourth quarter and fiscal year 2022 results,” 2023. Available at criteo.investorroom.com.
- [3] U. Riswanto, “K-means clustering for anomaly detection,” 2023. Available at medium.com.
- [4] C. Maklin, “Isolation forest,” 2022. Available at medium.com.
- [5] Akshara.416, “Isolation forest,” 2024. Available at analyticsvidhya.com.
- [6] A. GrabNGoInfo, “One-class svm for anomaly detection,” 2022. Available at medium.com.
- [7] G. Pierobon, “K-nearest neighbors (knn) for anomaly detection,” 2023. Available at medium.com.
- [8] D. Deng, “DbSCAN clustering algorithm based on density,” in *2020 7th International Forum on Electrical Engineering and Automation (IFEEA)*, pp. 949–953, 2020.
- [9] K. Khan, S. U. Rehman, K. Aziz, S. Fong, and S. Sarasvady, “DbSCAN: Past, present and future,” in *The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*, pp. 232–238, 2014.
- [10] Y. Jiang, C. Kang, Y. Shen, T. Huang, and G. Zhai, “Research on argo data anomaly detection based on improved dbSCAN algorithm,” in *Wireless Sensor Networks* (H. Ma, X. Wang, L. Cheng, L. Cui, L. Liu, and A. Zeng, eds.), (Singapore), pp. 44–54, Springer Nature Singapore, 2022.

¹⁵The CPC and commission rates were chosen from online sources and are on the conservative side; the most often-cited range for the CPC was \$0.1-\$1.0 and for the commission 20%-30%.

- [11] M.-K. Yoon, S. Mohan, J. Choi, and L. Sha, “Memory heat map: Anomaly detection in real-time embedded systems using memory behavior,” in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2015.
- [12] A. M. Ikotun, A. E. Ezugwu, L. Abualigah, B. Abuhaija, and J. Heming, “K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data,” *Information Sciences*, vol. 622, pp. 178–210, 2023.
- [13] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD’96, p. 226–231, AAAI Press, 1996.
- [14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.
- [15] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [16] K. Q. Weinberger, F. Sha, and L. K. Saul, “Learning a kernel matrix for nonlinear dimensionality reduction,” in *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML ’04, (New York, NY, USA), p. 106, Association for Computing Machinery, 2004.