Capstone Midterm Report: Heatmap Anomaly Detection

Xinyi Chen, Martin Fluder, Jean Law, Ling Lin, Yue Yin
Mentored by Jan Benzing and Matt Merenich
Criteo

March 14, 2024

Contents

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 ads that we need to click 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 potentially less surprisingly, Facebook's revenue stems to 98% 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 in 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 ad'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 ads non-functional.

Given that companies like Criteo display around 1 billion advertisements daily, the occurrence of broken banners represents a substantial revenue loss. The early (*i.e.* with minimal amounts of clicks) 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.

In the following report, we shall first set up the precise problem statement and analyze the datasets to which we have been given access by the Criteo team. We will then explore and describe the data and discuss various approaches for detecting anomalies within a given class of advertisement banners. Finally, we will extend our methods to the case of zero-shot (or one-shot) learning, which requires minimal training data for new banner types. We conclude with a brief discussion and future directions.

2 Broken Banner Detection in Online Advertisements

This capstone project aims to produce a generalizable, unsupervised, or self-supervised data processing pipeline and machine learning model that judiciously, quickly, and with as little data as possible classifies whether a particular advertisement banner is broken.

Advertisement banners come in various shapes and sizes, which are primarily dependent on the browser and device – phone, tablet, computer – in which a user is trying to access a specific webpage. We refer to Figure 1 for examples of advertisement banners.

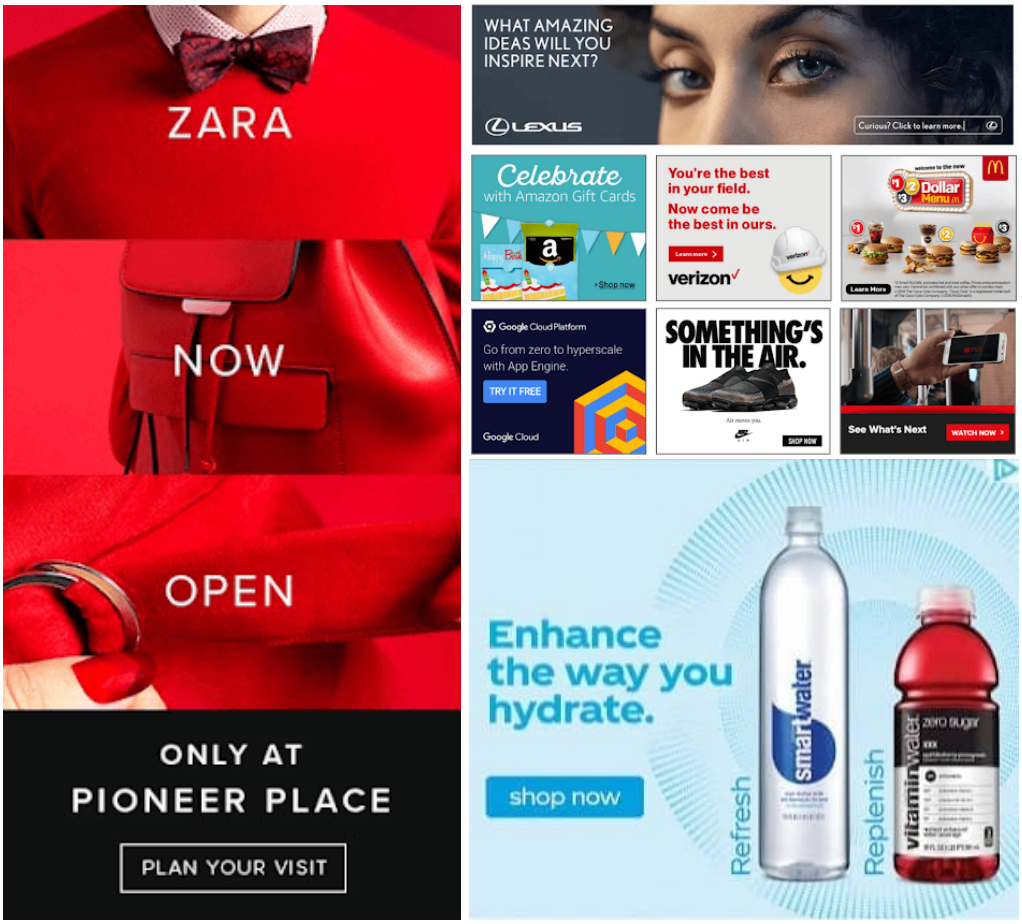

Figure 1: Four examples of advertisement banners, as displayed on webpages. Different ads have different form factors, and their products are arranged in different types of grids. The arrangement of "products" (or clickable links) within a banner is crucial for the detection of abnormal click behavior of users. For example, in the second banner on the right, we expect the average user to click in the center of either of the products, which yields a characteristic clicking pattern from which deviations can be classified as anomalies.

Criteo's banners are defined in terms of an underlying "grid_id", which determines the arrangement of different products or clickable links within a given banner and specifically affects the characteristic click pattern of our click-heatmaps. We refer to Figure 2 for Criteo's types and definitions of grid layouts.

Additionally, banners are characterized by "domain_id", which corresponds to the underlying combination of grid_id together with specific products, which will be deployed in various places. A certain domain_id can be broken for a variety of reasons, and it is the goal of this project to determine as soon as possible when this is the case, based on the user's click behavior, together with additional metrics of the banner.

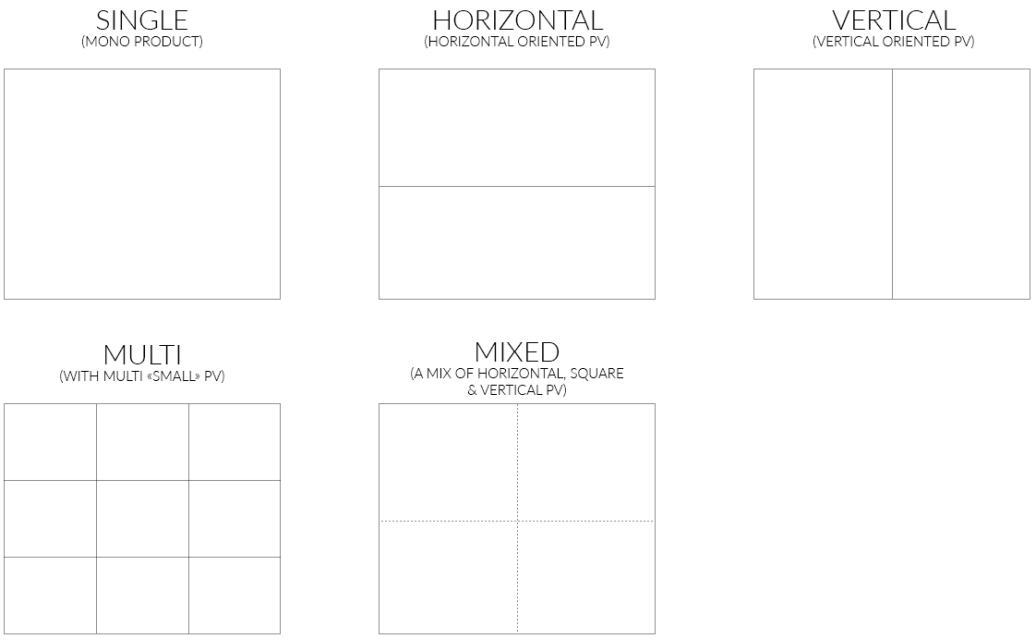

Figure 2: Grid layout for Criteo's advertisement banners. Our initial dataset consists of a mixed banner type with three products and a vertical arrangement with three products.

Our final product should include the following characteristics:

1. Generalizable to unseen banner types using *zero-shot learning*.
2. Handle class imbalance: less than 10% of all banners are broken.
3. Detect whether a banner is broken early, for instance, when it has less than 200 clicks, using metrics and data augmentation.
4. Lightweight and efficiently implementable so as to be able to scale to millions of banners daily. This is important because the broken banner detection algorithm is run every 24 hours and

hence would need to process on average around 5 million advertisements at each run.

2.1 Data

Criteo provided this project's data, consisting of two comprehensive datasets. The first – called *heatmap dataset* – consists of the breakdown of users' click behavior by grid_id and domain_id, while the second dataset – called *performance metric dataset* is a breakdown of other metrics of these banners, which we shall explain in the following.

Heatmap Dataset

This dataset consists of an aggregation of users' clicks acquired over a two-month period by the Criteo team and provided to us without any cleaning applied. Each click on a given banner corresponds to a tuple in this dataset and is uniquely identified by the key (domain, grid_id, click_x,click_y), in which click_x,click_y correspond to the location of the pixel that is being clicked on. More precisely, the dataset consists of the following attributes:

1. Domain: identifier for domain id.
2. Grid_id: identifier for grid, related to specific layouts and product placements.
3. Click_x: x-coordinate of pixel being clicked on.
4. Click_y: y-coordinate of pixel being clicked on.
5. Display_height, display_width: dimensions of the banner.
6. Clicks: recorded number of clicks on this specific pixel.

The display height and width of all the banners in this particular dataset is 250×300. We are given a total of 26 million clicks, each belonging to one of two grid_ids, 333519 and 333346 – *i.e.* arrangements of products within a banner (refer to Figure 2 for all possible layouts) – and roughly 800 domains for each grid_id. Both of these grid_ids display different characteristic click behavior. In Figure 3, we display all clicks associated to a given grid_id aggregated into a single click heatmap. We observe that, according to Criteo's grid layout definitions in Figure 2, the grid_id 333519 is of "mixed" type consisting of three products, while grid_id 333346 is of "horizontal" type also containing three products.

For each of the two grid_ids, we were given roughly 800 domain_ids, of which we have identified approximately 6.5% and 13.5% as broken, respectively. We refer to Section 2.2 for more details in how we obtained the "ground truth."

We have found that each heatmap domain we have been given access to has at least 200 clicks. This is in order to have a meaningful chance of identifying broken banners. The distribution of the number of clicks per domain is shown in Figure 4. There is a clear skew towards fewer clicks, which mirrors the real-life situation where we want to identify broken banners as early as possible. In the following, we shall employ various data enhancement strategies to judiciously identify broken banners.

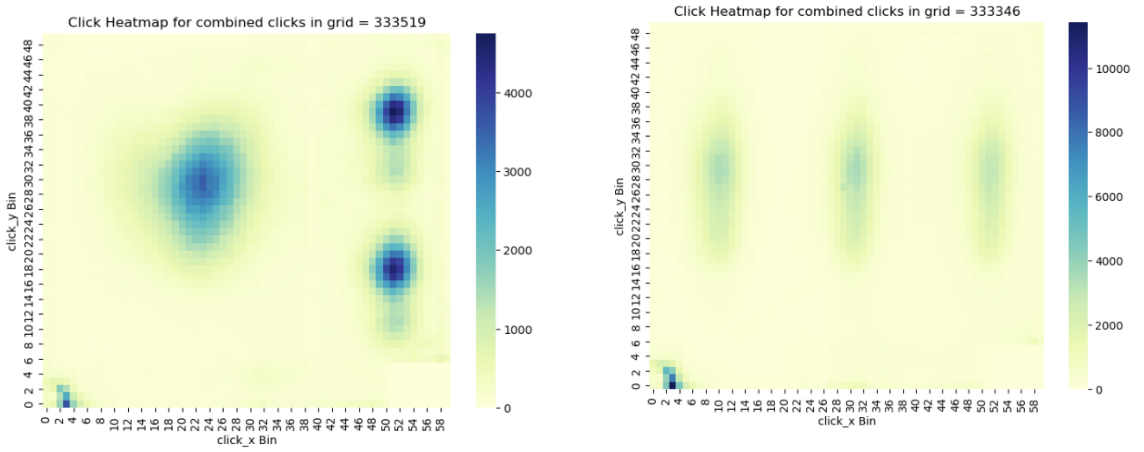

Figure 3: Aggregation of all clicks by grid_id. We see that both grid_ids display a different characteristic pattern based on their layout type. We notice an aggregation of clicks on the bottom left, corresponding to the "x"-button to click out of the advertisement. Additionally, the bottom right corner is devoid of clicks, which is something we attest to an overlapping/non-clickable area.

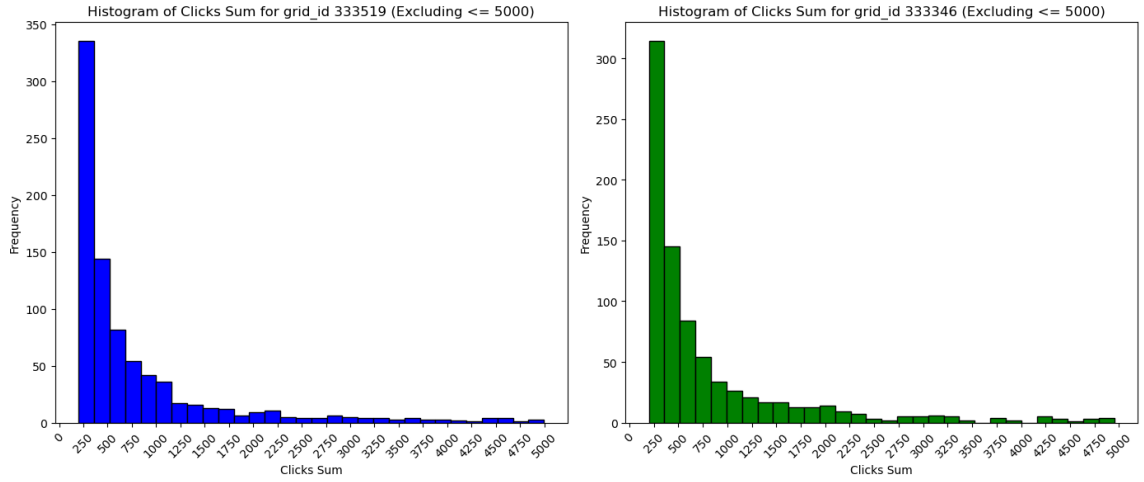

Figure 4: Histogram of the number of clicks for each domain_ids in the heatmap dataset. There is a skew towards 200-500 clicks, which aligns with the idea of determining a broken banner as quickly as possible, *i.e.* with as few clicks as possible.

Performance Metrics Dataset

The performance metrics dataset consists of metrics data for a superset of domains as the heatmap dataset for the same two grid_ids. The Criteo team has acquired this data for a subset of the two months over which they acquired the heatmap dataset, as it is generally easier and faster

to get this type data. We have been provided the raw data without any cleaning applied. The dataset contains the following information.

1. Domain, grid_id: identifier for domain and grid.
2. Webview_height, webview_width: dimensions of the banner.
3. Displays: number of times the ad banner was displayed.
4. Clicks: number of times users clicked on the banner.[1]
5. Landed_clicks: number of clicks that successfully led to a landing page.
6. Non_bounced_clicks: number of clicks where the user did not immediately leave the site.
7. Closing_events: number of times users closed the ad.
8. Avg_last_second_framerate: average framerate in the last second.
9. Sov_short_ttc: difference between the clicks and display timestamps (ttc = time to click).
10. Sov_short_ttc_global: difference between the clicks and display timestamps across all domains.
11. 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*.

There are roughly 300 missing values in the rows of this dataset, which we shall address in the following.

2.2 Generating Ground Truth Labels

To assess our models, we had to first determine the ground truth of broken banners. To do so, we defined a "clearly broken banner set" as the set of banners which are broken under the following steps of data enhancement:

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 and perturb it with Gaussian noise of fixed standard deviation) to enhance each heatmap to 5000 clicks.
3. If the triangle, line structure for grid_id 333510, 333346, respectively, is not clearly identifiable even with this noisy bootstrap enhancement, we define the heatmap as broken.
4. We do not care about the rest of the heatmap if this pattern is clearly defined.

[1]Notice that this number is different from the sum of the number of clicks recorded for the heatmap dataset because they were recorded over a different time span.

Conditions 3 and 4 implies that we identify a banner as "clearly broken", even though users exclusively clicked on the "x" to close the advertisement. There are many such examples, and while the banners might not inherently be broken, from a business perspective, it makes sense to define them as anomalous since they are ineffective in getting users to click on the products. In Figure 5, we show a representative sample of banners we identified as clearly broken. We can contrast them with the representative heatmaps in Figure 3, which clearly display the required triangle, horizontally aligned structure, respectively.

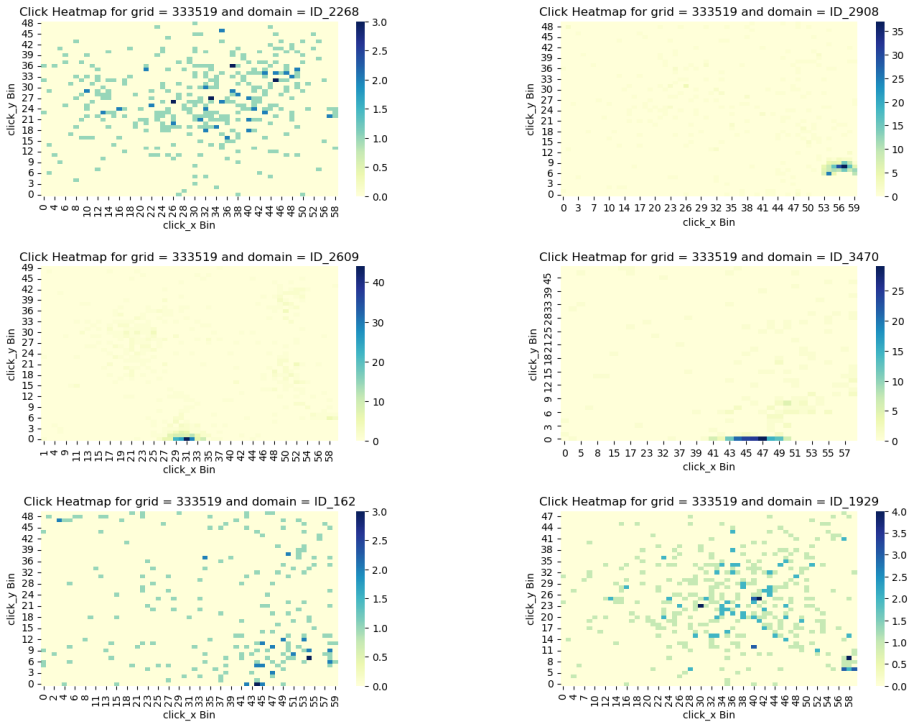

Figure 5: Examples of "clearly broken banners." We observe that for none of these, the triangle/line structure of products is visible, and the pattern of clicks is either random or localized at the corners of the banner. Even if these banners are not broken, they are clearly ineffective in getting users to click on the products and, therefore, should be fixed.

We have used various techniques to narrow down the search space and identify the clearly broken banners, but effectively, it was an effort primarily by hand. By doing so, we have identified 57 clearly broken domains out of 872 for grid_id 333519. This corresponds to roughly 6.5%, which is in line with the numbers provided to us by the Criteo team (roughly 5-10% of banners are broken). For the grid_id 333346, we have identified almost 113 clearly broken grids, corresponding to 13.5%. This is more than expected, and we (including our mentors) currently do not know why our data shows a significantly larger number of broken banners than expected.

2.3 Existing Work and Technology

We have had inspiration from different state-of-the-art techniques across the fields of heatmap analysis, computer vision and anomaly detection to derive a comprehensive approach to our problem.

From the field of anomaly detection, we mainly researched ideas of unsupervised learning algorithms, including K-Means clustering [3], KNN [4], one-class SVM [5], Isolation Forest [6, 7], and DBSCAN [8, 9, 10]. Each of these techniques is quite well-known for its capability to identify outliers without the need for labeled data. The application of these algorithms in various domains, from network security to environmental monitoring, shows their versatility and effectiveness in detecting deviations from established patterns [11]. Using such techniques, our project aligns with the current anomaly detection research.

Combining heatmap analysis with anomaly detection techniques forms the core of our solutions since we have the banner data as heatmaps. While direct studies in heatmap anomaly detection are less common, the application of algorithms like K-Means clustering and DBSCAN in related areas provides a solid foundation for our approach. These algorithms' ability to discern subtle anomalies within complex data structures is particularly relevant for analyzing banner heatmaps, where anomalies may manifest as irregular patterns or unexpected variations. The principles derived from ad fraud detection and related fields offer a contextual understanding. Our project extends different principles, employing advanced unsupervised learning techniques to effectively identify and isolate broken or malfunctioning banners.

These diverse areas of research inspire our approaches, and we want to integrate state-of-the-art techniques to solve the unique challenges of broken banner detection.

Criteo's current system for anomalous and broken banner detection solely relies on explicit thresholds of a given banner's performance metrics. A significant advancement of their system would be the inclusion and addition of techniques that deal with heatmaps and can potentially extend to examples with only a small number of clicks.

3 Unsupervised Learning Methods

In this section, we focus on unsupervised learning approaches for identifying broken advertisement banners. These methods perform inherently well but require a non-insignificant amount of data per grid_id to reliably identify broken heatmaps. We first introduce our overall data preprocessing pipeline, which includes principal component analysis to reduce the dimensionality of the click-heatmap vectors. We shall briefly outline why PCA is important and powerful by analyzing the first few PCA vectors. We then discuss various clustering methods solely based on the heatmap dataset and conclude by describing clustering methods for the combined datasets.

In the future, we shall turn our attention towards "zero-shot" approaches, that generalize to grid_ids with none or very little data. Such methods would be inherently crucial for their business, as grid_ids/grid arrangements change often, and it would be beneficial to not have to train a model on each new id.

3.1 Data Pre-processing and Principal Component Analysis (PCA)

Our heatmap datasets contain pixel-level clicks for each grid_id and domain combinations. To reduce the dimensionality of our vectors, we employ a binning 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 and provides figures such as Figure 3 and Figure 5.

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. While we have employed and studied different non-linear techniques (*e.g.* Maximum Variance Unfolding (MVU) [12]), the simplicity and power of principal component analysis (PCA) convinced us to mostly focus on this technique. To get consistent and comparable results for PCA, we must carefully normalize each heatmap vector to have equal number of clicks. We achieve this by normalizing the i-th heatmap vector as

$$h_i \;\to\; \widetilde{h}_i \;=\; \frac{h_i}{\text{total number of clicks in } i^{\text{th}} \text{ heatmap}}$$

before applying the PCA transformation to the matrix $H = \{\widetilde{h}_i\}_{i=1}^{N}$.[2]

In Figure 9, we show the two-dimensional plot of the first and second PCA vectors for both grid_ids together with their ground truth labels. It is apparent that the first PCA vector seems to capture the "brokenness" of click heatmaps consistently. Since the first PCA vector appears to be a good feature for capturing whether a banner is broken, we further investigate the coefficients of each 5×5 bucket of the first and second PCA vectors and display them as a heatmap in Figure 7. As shown in that figure, the first PCA vector in both cases captures the "noise," *i.e.* the clicks outside the three central clusters, in both grid_ids. Since proportionally, many clicks in these outer regions of the banner indicate that the banner might be broken, larger values of the first PCA seem to correspond to a higher likelihood of the banner being broken.

From Figure 7, we further observe that the second PCA vector also seems to capture noise in the case of grid_id 333346, while for grid_id 333519, it seems to pick up the central clusters and, therefore, a direction indicating non-broken banners. This intuition is in accordance with the two-dimensional PCA plot in Figure 9.

To streamline the pre-processing of our heatmap dataset, we created 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). Our first step expanded the dataset, so each row corresponds to an individual click. We then divided the *click_x* and *click_y* columns into 60 and 50 bins, respectively, to discretize the click coordinates. Following this, we summed the clicks within each bin and included entries for empty bins to simplify vector creation. Subsequently, we reformatted the aggregated data into a broader structure where each row represents a domain of the specified grid_id, detailing the click distribution across each bin. To ensure comparability across heatmaps, we normalized the data so that the total click count for each heatmap equals one. After normalization, we applied PCA based on the specified component

[2]Not normalizing the vectors by click numbers, results in much less expressive PCA components, since the PCA vectors pick up the number of clicks in a specific heatmap as a feature. However, this is a feature we are not interested in.

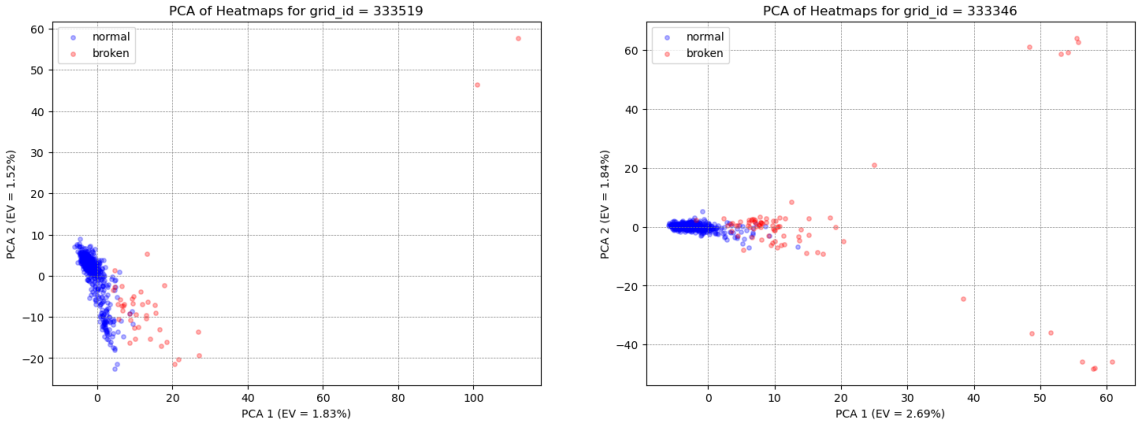

Figure 6: Plots of the first and second PCA vectors for our click heatmaps for both grid_ids. Next to the PCA vectors, we display the explained variance (EV) for each vector for our dataset.

count, integrated the ground truth labels for each domain, and stored the pre-processed data in a temporary directory for subsequent analysis.

3.2 Heatmap Clustering

We now turn towards the various clustering methods we employed to determine whether a heatmap is broken.

Within the domain of broken banner detection, it is essential to select appropriate analytic methodologies to establish reliable detection systems. In our research, we delve into the application of various machine learning approaches for broken banner detection. Specifically, we will focus on six distinct methodologies to serve as baseline models: Click-clustering, K-Nearest Neighbors (KNN), K-Means Clustering, Isolation Forest, One-Class SVM, and DBScan. Each approach provides specific perspectives and mechanisms for identifying irregularities and anomalies within the datasets.

In our analysis, we performed a development and test data split on the heatmap dataset to help evaluate the selected methods unbiasedly. Then, we conducted a hyperparameter tuning on the training dataset to optimize the performance for each model. Subsequently, these optimized models are applied to test data. Performance metrics such as accuracy, precision, recall, and F1-score are utilized to check the effectiveness of the models. We also generated predicted labels to further aggregate across all heatmap clustering methods.

3.2.1 Click-clustering and Hypothesis Testing

The "click-clustering" approach is based on a statistical idea; we can create a distribution over the x- and y-axis of the banner empirically derived from all the aggregated clicks. We can use a chi-squared test to compare whether this empirical distribution corresponds to the distribution of clicks

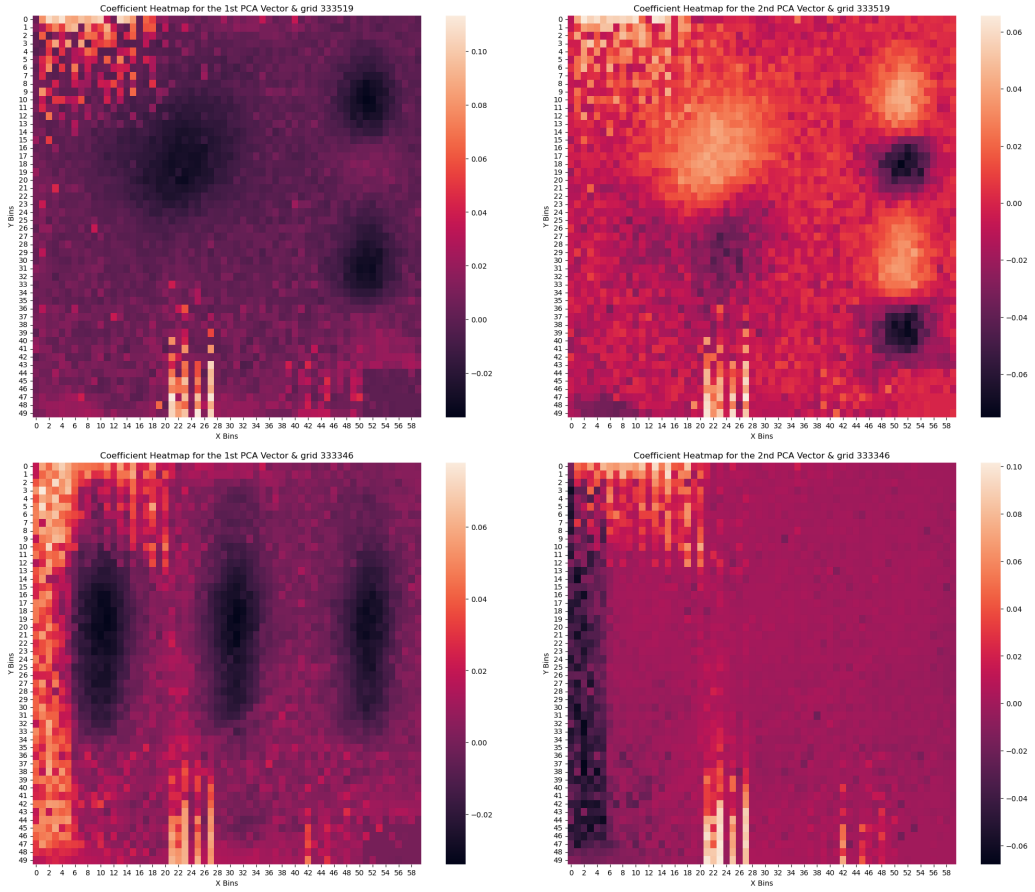

Figure 7: Heatmap plots of the coefficients of the first and second PCA components for both grid_ids 333519 (top row) and 333346 (bottom row). For both grid_ids, the first PCA component picks up the noise outside the three main click clusters.

of a specific domain heatmap. Alternatively, we can compute the probability for the distribution of a specific domain heatmap given the empirical overall distribution and classify the heatmap as anomalous if the probability is below a certain threshold.

These two methods seem like a natural starting point, but perform relatively poorly. However, a small change to the approach performs much better. Here is the description of the algorithm:

1. Bootstrap 100,000 clicks from the fully aggregated dataset (filtered by grid_id).

2. Run a clustering algorithm, DBSCAN (see section 3.2.6), with certain hyperparameters on these 100,000 clicks and remove any potential clusters at the bottom of the picture. See Figure 8 for an example of the identified clusters.

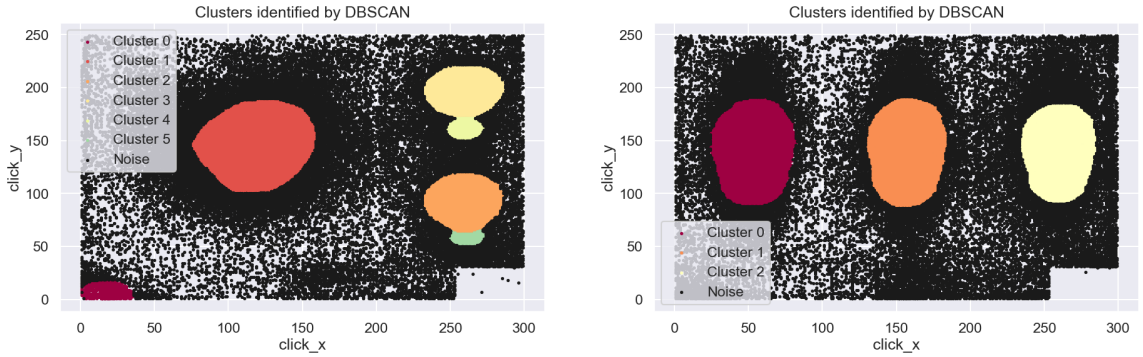

Figure 8: DBSCAN's cluster identification for grid_id 333519 (left) and 333346 (right). Given this clustering, in each heatmap, we count the ratio of clicks that lie within any of the central clusters (we remove the ones below 25 pixels in the y-axis) over the total number of clicks and based on whether this percentage is above or below a certain threshold decide on whether the heatmap is broken.

3. For a given (noisy bootstrap enhanced) heatmap, assign to each click a cluster (or noise) label based on the DBSCAN nearest neighbor.

4. Run a hypothesis test: H_0 : percentage of noise points in training data $<$ percentage of noise in the current heatmap. If the p-value is below certain threshold, the heatmap is classified as broken, else not.[3]

This algorithm performs very well, and the hyperparameters and p-value/noise percentage threshold transfer from one grid_id to the other, suggesting that the method could generalize to any grid_id given a representative 100,000 clicks on which to create the DBSCAN clusters. We have collected our results in Table 1.[4]

3.2.2 K-Means

K-means clustering, an unsupervised machine learning algorithm, categorizes datasets into K groups based on the proximity to their means. It starts by randomly choosing K centroids and, through iterative adjustments, assigns each data point to the nearest cluster, refining the process until the internal distance within each cluster reaches a minimum. In anomaly detection, this algorithm is adept at identifying outliers, which are data points that either belong to smaller clusters or are significantly distant from their centroids.

Our selection of K-means as a baseline model was motivated by its straightforward implementation, interpretability, and scalability, allowing for efficient processing of large datasets. Nonetheless,

[3]This approach corresponds to a Binomial hypothesis test. Since we upsample consistently to the same number of clicks for each heatmap, the techniques is actually equivalent to comparing the noise percentage against a certain threshold.

[4]Notice that we work with a noise threshold percentage rather than a p-value threshold. However, since we up-sample to 5,000 clicks for each heatmap, both thresholding methods are equivalent.

K-Means also has some limitations. For example, it assumes the clusters are spherical and have equal variances. The initial placement of centroids also affects the final clustering results.

Initially, applying K-means directly to our heatmap dataset and labeling the smaller clusters as anomalies did not yield satisfactory results upon visual inspection of the generated heatmaps. This led us to integrate Principal Component Analysis (PCA) to reduce the dataset to two dimensions before applying K-means, significantly enhancing the accuracy of our anomaly detection, as confirmed by heatmap inspections and comparisons with ground truth labels derived from DBSCAN.

In the process of optimizing our model, we focused on tuning hyperparameters with the aim of bolstering performance on unseen data. Although we explored adjusting the PCA dimensions and the number of clusters (K), our findings indicated that variations in K minimally impacted the F1 score, leading us to maintain K=2 for its simplicity and direct relevance to our binary classification task (broken vs. non-broken). For grid_id 333346, a PCA reduction to 54 dimensions achieved an F1 score of 0.16, indicating room for improvement. For grid_id 333519, an optimal PCA dimension of 86 resulted in a robust F1 score of 0.83.

To further refine our K-means model, future efforts could explore enhancing anomaly detection accuracy by focusing on domain ids significantly distant from cluster centroids and then examining the effect of different K values on model performance, especially to potentially improve outcomes for grid_id 333346.

3.2.3 Isolation Forest

The Isolation Forest algorithm excels in anomaly detection within high-dimensional datasets, making it ideal for our heatmap data analysis. This unsupervised learning method stands out because it does not depend on distance or density measures to identify anomalies. Instead, it utilizes decision trees to isolate outliers, effectively distinguishing them based on the minimal splits required. This characteristic is especially useful for our project, as anomalies in banner heatmaps are typically rare and significantly different from normal data points. The scalability and computational efficiency of the Isolation Forest are crucial for handling the large datasets common in digital advertising, thereby aligning perfectly with our project's needs.

For our model, fine-tuning was essential to adapt it to the specifics of our dataset, which included different Grid IDs. We focused on tuning two main hyperparameters: the number of PCA dimensions and the number of estimators in the Isolation Forest. These parameters were critical in ensuring a robust model that minimizes the impact of high dimensionality, a common challenge in our data. Given the imbalanced nature of our dataset, with anomalies constituting less than 10% of the data, we opted for the F-1 Score as our performance metric. Through careful tuning on the training set, we identified optimal hyperparameters for each grid ID. For grid ID 33346, we found that 50 estimators and 6 PCA dimensions yielded the best results. Similarly, for grid ID 333519, 100 estimators and 10 PCA dimensions were optimal. The performance outcomes of these configurations are detailed in Tables 1 and 2.

Future explorations with the Isolation Forest may delve deeper into analyzing true positives and true negatives, possibly considering the format variations of the banners to enhance detection accuracy.

3.2.4 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. This approach is especially relevant in scenarios where labeled data is scarce or imbalanced, making traditional supervised learning methods less effective.

The first step involved thorough preprocessing of the dataset. This included cleaning, normalizing, and reducing its dimensionality using PCA. Such preprocessing was vital to ensure the data was in an optimal state for modeling, aiding the OCSVM in capturing essential patterns. We selected the One-Class SVM model for its proven track record in anomaly detection. The configuration of the model was a critical phase where parameters were chosen to best suit our data characteristics. We chose for the kernel='rbf' due to its ability to manage non-linear data relationships, which are common post-PCA. We chose gamma='auto' to allow the model to adjust this parameter based on the features' number, aiming for a balanced decision boundary definition. After configuring the model, we trained it on the PCA-reduced dataset. Our focus was on distinguishing normal data points from potential anomalies. We then evaluated the model using metrics such as True Positive Rate (TPR), True Negative Rate (TNR) and F-1 score. These metrics shed light on the model's accuracy in anomaly detection and its ability to minimize false positives. To get the best model, we chose to implement hyperparameter tunning for PCA dimensions.

While the model demonstrated an ability to reduce false positives, it showed a need for improvement in its detection accuracy, as suggested by the TPR. This underscores the complexity of anomaly detection in high-dimensional data and opens avenues for exploring more sophisticated models, feature engineering techniques, and evaluation metrics in future work.

3.2.5 KNN

The k-Nearest Neighbors (KNN) algorithm serves as a foundational approach and baseline model to handle the classification and unsupervised anomaly detection tasks. It classifies a new data point by examining the class levels of its k nearest neighbors in the feature space. As a result, similar data points tend to belong to the same class. 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.

KNN is particularly useful for its simplicity and effectiveness for identifying outliers within high-dimensional datasets. In our research, we enhanced its performance by integrating it with Principal Component Analysis (PCA) for dimensionality reduction, so it becomes more efficient in handling large and complex datasets. To optimize the model performance, we conducted hyperparameter tuning by adjusting parameters such as the number of nearest neighbors, threshold, and PCA dimension.

For grid_id 333346, we found that reducing the dimensionality to 6 using PCA effectively captures the most important variance in the data while reducing noise and computational complexity. Employing 10 nearest neighbors provides a more robust estimation of the local density around each point. And setting a threshold of 90% on the anomaly scores provides a good trade-off between

identifying anomalies and minimizing false positives. In this case, we achieved a F1 score of 0.8. Similarly, for grid_id 333519, employing a 10 PCA dimension, 5 nearest neighbors and 95% threshold provides the best training performance with a F1 score of 0.85.

Subsequently, we evaluated the model with best hyperparameters on the test dataset, where we observed promising performance metrics. For grid_id 333346, the F1 score obtained was 0.83, and for grid_id 333519, it was 0.71. The results suggest that the KNN model performs quite well across the evaluated metrics. The predicted labels from KNN are then utilized for further computations on finding the best combination of the 5 models.

3.2.6 DBSCAN

Rather than proximity as in KNN, DBSCAN is a density-based clustering algorithm. It groups together data points that are closely packed, while marking points that are in low-density regions as outliers or noise. 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. In broken banner detection, DBSCAN can be used to identify clusters of banner images that exhibit similar characteristics and to highlight potential anomalies indicative of broken banners.

In our research, we also enhanced the DBSCAN algorithm's performance by integrating it with Principal Component Analysis (PCA) for dimensionality reduction. For grid_id 333346, employing a PCA dimension of 12, an epsilon value of 10, and a minimum sample size of 10 yielded notable results. The F1 score obtained was 0.94, indicating robust performance in both precision and recall aspects. This configuration showcases the effectiveness of DBSCAN in capturing dense regions within the data space, even in a high dimensionality. Similarly, for grid_id 333519, employing a PCA dimension of 8, with epsilon and minimum sample size set to 10, resulted in a commendable F1 score of 0.93.

Then, we evaluated the model on the test dataset. For grid_id 333346, the F1 score obtained was 0.85, and for grid_id 333519, it was 0.77. The results obtained show that the configuration of DBSCAN is promising, and it effectively identified anomalies and clustering data points. It served as a robust foundation for further analyses and model evaluation within our research framework.

3.2.7 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. Specifically, this approach achieved an F1 score of 0.97 for grid_id 333346 and 0.87 for grid_id 333519.

By using these techniques as baselines, we can check their efficacy in capturing broken banners with a comparative analysis on their performance. Through this comprehensive examination, we

have laid a solid groundwork for future investigations into more advanced detection methods to enhance the reliability and efficacy of broken banner detection systems.

3.3 Combined Heatmap and Performance Metric Dataset

Thus far, we have mostly focused on the heatmap dataset. However, it is crucial to investigate whether combining the click-based data from the heatmap dataset together with the performance metrics dataset can lead to improved clustering results. To do so, we first have to find a meaningful way to combine the two datasets.

First, let us focus solely on the performance metrics dataset and explore whether there is a supervised learning method which performs well exclusively on the metrics data given our ground truth developed on the heatmap dataset. We employed various techniques to balance the dataset (up/downsampling), add multiple interaction terms, remove multicollinear features, and train several classifiers on the labeled data. However, they perform only somewhat comparable to our pure heatmap methods.

Therefore, we decided to combine the two datasets to capture more pertinent features. We do so as follows:

1. PCA on the heatmap dataset down to say 200 components.[5]
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 $\widetilde{\text{PCA}}$. Figure 9 shows the first and second $\widetilde{\text{PCA}}$ directions. The unbroken banners seem clustered in the center, while the broken banners are clearly "noise" away from these central clusters. While the $\widetilde{\text{PCA}}$-pictures look promising, running the same type of clustering hyperparameter tuning approach as for the heatmap-only dataset leads to (significantly) worse-performing models. We are not entirely sure why this is the case. Still, we believe that while the heatmap PCA vectors are powerful in capturing noise outside of the central clusters, the inclusion of the performance metric dataset seems to significantly affect the variance of the $\widetilde{\text{PCA}}$-vectors and therefore pick directions that are "worse" at distinguishing noise versus clustering.

4 Zero/Few-shot Learning Approaches (future directions)

So far, we have exclusively dealt with anomaly detection methods that rely on a solid amount of data; for example, to identify a heatmap vector as noise, we require an existing cluster of non-noise heatmap vectors (domains). Putting any of these methods into production would require a

[5]The 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 sitll inferior to the heatmap-only clustering.

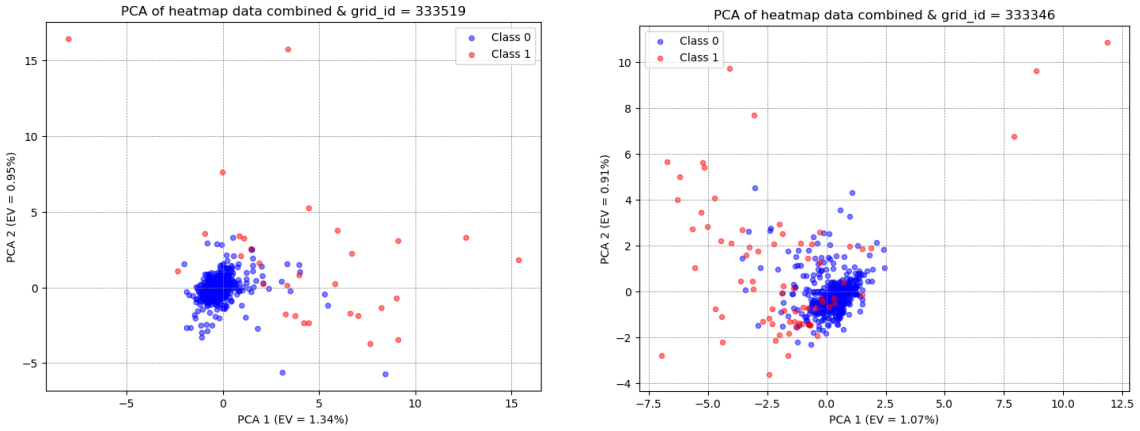

Figure 9: Plot of the first and second $\widetilde{\text{PCA}}$ directions from the combined (PCA-ed) heatmap and performance metric dataset.

separate algorithm for each grid_id. It is technically feasible but would further require the Criteo team to "train" a new model for each new grid_id they introduce. It is, therefore, essential for them to explore global methods that judiciously identify broken banners (almost) independent of the underlying grid_id (or grid-arrangement).

We are currently working on various approaches and only briefly mention some directions we are currently exploring.

- We can use pre-trained ResNet-50 or ViT models as feature extractors. We feed them not a picture but a heatmap and extract the features (the vectors prior to the classifying layer) they generate from the heatmap. Given that these models are very good at identifying patterns and different types of images, we imagine that (like humans) they can distinguish broken from non-broken banners. Some preliminary results suggest that using cosine similarity for both pre-trained ResNet and ViT models leads to relatively clearly separable clusters of broken versus non-broken banners.

- We can go further with this intuition and fine-tune the ResNet-50 model on synthetic images and tasks. It is straightforward to generate heatmaps with various types of clusters, and we can consider training the ResNet model on identifying, say, the number or location of clusters in a heatmap. Intuitively, this should refine and improve its ability to detect and distinguish broken from non-broken banners. Some preliminary results suggest that fine-tuning such synthetic tasks/data helps pull apart broken versus non-broken banners, and we are currently working on combining the features of various fine-tuned models into a single robust model.

- Finally, we are currently thinking about implementing a "contrastive learning model." The idea would be to train an encoder-decoder model that recreates a heatmap while simultaneously distinguishing the input from a *different* heatmap (contrastive learning). That way, we can generate a set of features in a compressed vector space that reliably distinguishes broken

from non-broken banners.

5 Discussion and Future Directions

In section 4, we discuss several directions we are currently exploring that we believe could generalize to unseen grid_ids without additional training and with little supervision. We already have some promising results with the pre-trained and fine-tuned ResNet and ViT models. We intend to make progress on the contrastive learning approach and explore methods to combine these models into a single zero-shot pipeline.

Our literature searches for similar problems have yet to turn up much information on the state-of-the-art – if there is a state-of-the-art. Therefore, we have mainly employed our understanding in developing methods and will further investigate the current computer vision research literature to possibly find ideas to apply to our problem.

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.

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] G. Pierobon, "K-nearest neighbors (knn) for anomaly detection," 2023. Available at medium.com.

[5] A. GrabNGoInfo, "One-class svm for anomaly detection," 2022. Available at medium.com.

[6] C. Maklin, "Isolation forest," 2022. Available at medium.com.

[7] Akshara_416, "Isolation forest," 2024. Available at analyticsvidhya.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.

[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] 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.

Method	Grid ID	Hyperparameters	Train (F1 Score)	Test (F1 Score)
1. Click-Clustering (DBSCAN)	333346	epsilon = 0.13 MinSample = 1000 Noise-% threshold = 57%	0.93	0.93
1. Click-Clustering (DBSCAN)	333519	epsilon = 0.13 MinSample = 100 Noise-% threshold = 68%	0.91	0.90
2. K-Means	333346	PCA dimension = 54 $k = 2$	0.34	0.16
2. K-Means	333519	PCA dimension = 86 $k = 2$	0.93	0.83
3. Isolation Forest	333346	PCA dimension = 6 n estimator = 50	0.90	0.96
3. Isolation Forest	333519	PCA dimension = 10 n estimator = 100	0.90	0.60
4. SVM	333346	PCA dimension = 33	0.43	0.30
4. SVM	333519	PCA dimension = 182	0.17	0.15
5. KNN	333346	PCA dimension = 6 $k = 10$ threshold = 90%	0.80	0.83
5. KNN	333519	PCA dimension = 10 $k = 5$ threshold = 95%	0.85	0.71
6. DBSCAN	333346	PCA dimension = 12 epsilon = 10 MinSample = 10	0.94	0.85
6. DBSCAN	333519	PCA dimension = 8 epsilon = 10 MinSample = 10	0.93	0.77
7. Combined	333346	All 5 models combined	N/A	0.97
7. Combined	333519	All 5 models combined	N/A	0.87

Table 1: Summary table of model performance for both Grid IDs using several types of clustering algorithms.