Capstone Project: Heatmap Anomaly Detection

Week 5 Progress Report

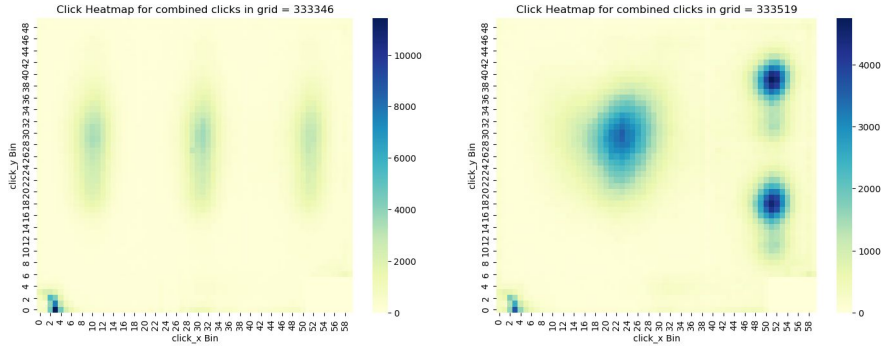
Agenda for Today

- Ground truth labels
- Heatmap clustering using DBSCAN.
- Isolation Forest Method
- First ideas for metrics
- Questions

Baseline:

Definition of “clearly broken grid:”

Triangle/line structure not visible even with noisy bootstrap enhancement. We do not care about rest of heatmap as long as this pattern is clearly defined.

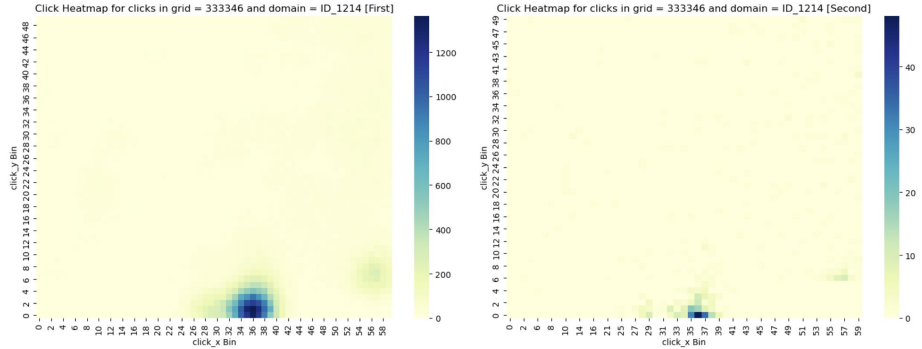


1. Created baseline of “clearly broken banners” for two grids:
   - “Triangle grid”: 57 clearly broken out of 872 → **~6.5%**
   - “Line grid”: 113 clearly broken out of 861 → **~13.5%**
   - Is it reasonable that there is such a discrepancy?

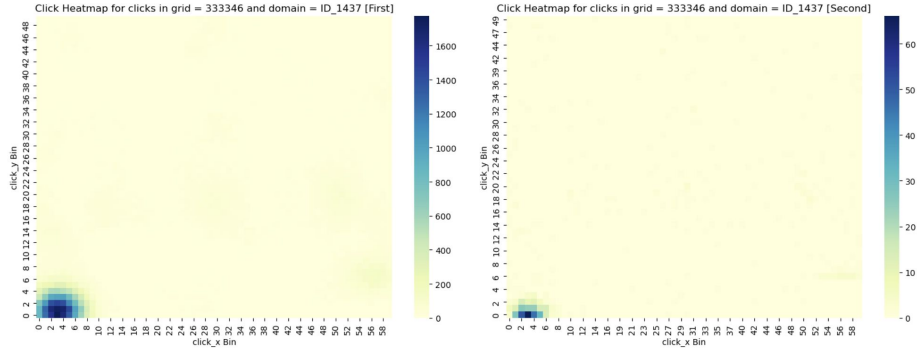
2. Added heatmap images for banners classified as “broken” to GitHub.

Baseline: Question

Current id: ID_1214:

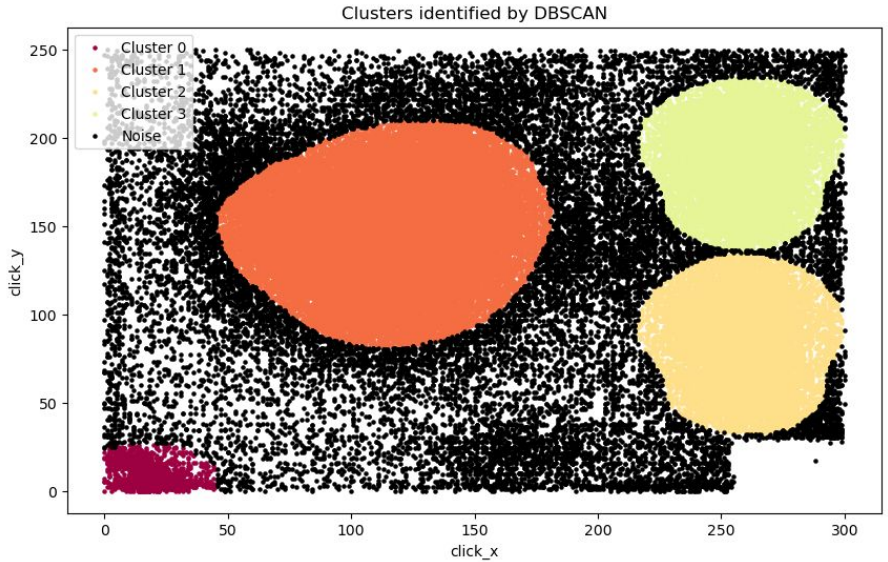


Current id: ID_1437:



- Many examples with clicks on ‘x’ of banner but no “center clicks” (see example 2) → is this broken or just majority of users want to click out of ad?
- Similarly, what if the clicks are not localized at the corner? (see example 1)

Clustering approach of heatmap clicks:



Estimated number of clusters: 4
Pct of noise points: 0.173037

1. Bootstrap 100’000 clicks from fully aggregated dataset (filtered by grid_id).
2. Normalize (Standardize)
3. Run DBSCAN cluster with eps = .2 and min_samples = 1000
   → 4 clusters + noise.
4. For given (noisy bootstrap enhanced) domain, get 1-nn for each click in training data and select that label {0,1,2,3}
   a. If pct of points labelled as noise above a certain threshold → anomalous.
   b. Hypothesis testing: p_0 = pct of noise points in training data. H_0: p_0 < noise/total, H_A: p_0 >= noise/total → p-value larger than threshold (cannot reject null) → anomalous.

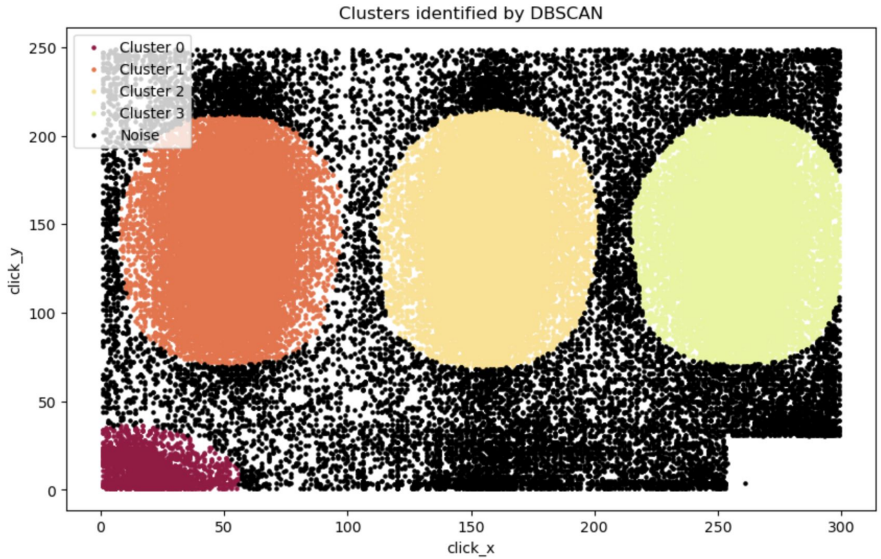
Clustering approach of heatmap clicks:

Pct of noise points: 0.222888

1. Bootstrap 100'000 clicks from fully aggregated dataset (filtered by grid_id).
2. Normalize (Standardize)
3. Run DBSCAN cluster with eps = .2 and min_samples = 1000
   → 4 clusters + noise.
4. For given (noisy bootstrap enhanced) domain, get 1-nn for each click in training data and select that label {0,1,2,3}
   a. If pct of points labelled as noise above a certain threshold → anomalous.
   b. Hypothesis testing: p_0 = pct of noise points in training data. H_0: p_0 < noise/total, H_A: p_0 >= noise/total → p-value larger than threshold (cannot reject null) → anomalous.

Upshot:

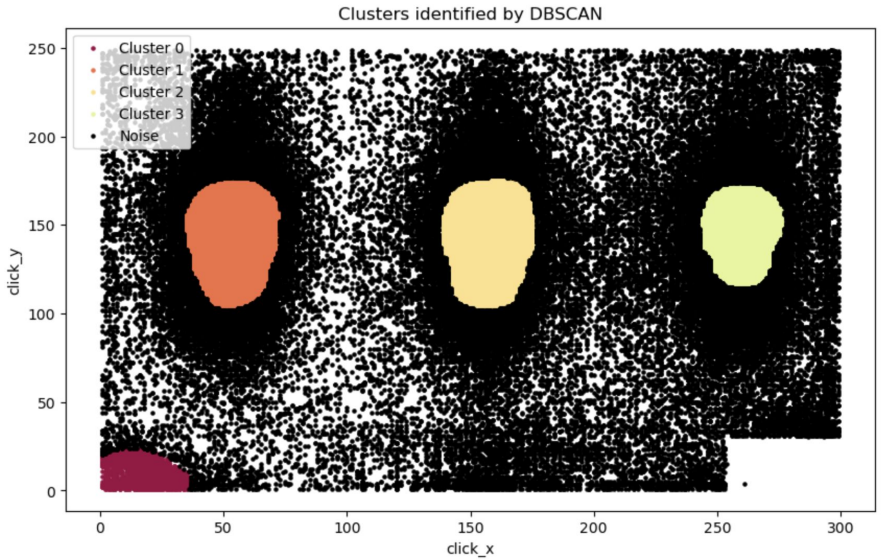
```
Estimated number of clusters: 4
Estimated number of noise points: 16706
Pct of noise points: 0.167060
```



- Change cluster regions, keep or remove cluster 3, Pick threshold → run grid search over these choices → confusion matrix

Upshot

```
Estimated number of clusters: 4
Estimated number of noise points: 53800
Pct of noise points: 0.538000
```



- Some examples:
  - ~50% TPR, ~100% TNR (eps = 0.12, thresh = 54%, 3rd not noise).
  - ~60% TPR, ~97% TNR (eps = 0.15, thresh = 32%, 3rd not noise).
  - ~53% TPR, ~99% TNR (eps = 0.18, thresh = 27%, 3rd not noise).
- Some remarks:
  - 3rd = noise performs badly
  -
- What should we optimize/prioritize on confusion matrix (positive = broken)?
  - TPR vs TNR?

Isolation Forest

- **Overview of Isolation Forest:**
  - An unsupervised, ensemble-based algorithm designed for anomaly detection.
  - Efficient with high-dimensional data, ideal for clickstream analysis.

- **Data and Preprocessing:**
  - Analyzed click data represented in a 50x60 bin grid, totaling 3000 dimensions.
  - Preprocessed to account for varying display sizes and click distributions.

- **Model Application:**
  - Applied separately to two types of grids (IDs: 333346 & 333519) to maintain contextual integrity.
  - Anomalies are instances with atypical click patterns, potentially indicating issues such as ad fraud or design flaws.
    - n_estimators: Robustness with 500 trees
    - contamination: Expected proportion of outliers (10%)
    - max_features: Balance between performance and computation
    - bootstrap: Randomness to prevent overfitting
    - random_state: Reproducibility of results

Isolation Forest Result:

Anomaly Result:

- **333346**: {'Anomalies': 114, 'Non-Anomalies': 1027}, 9.99%
- **333519**: {'Anomalies': 114, 'Non-Anomalies': 1027}, 9.99%

Hyperparameter: Contamination = 0.1

Overlap with Broken_Domain data, verified by human:

- **Total** Broken_Domian for **333519** count: 57

  %: 15.04424778761062 %

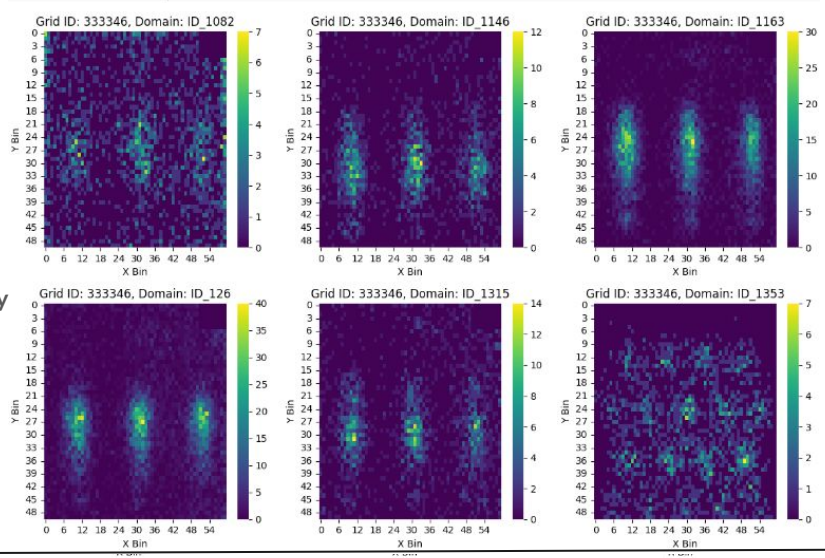
  Count: 17

  Total Broken_Domian for **333346** count: 113

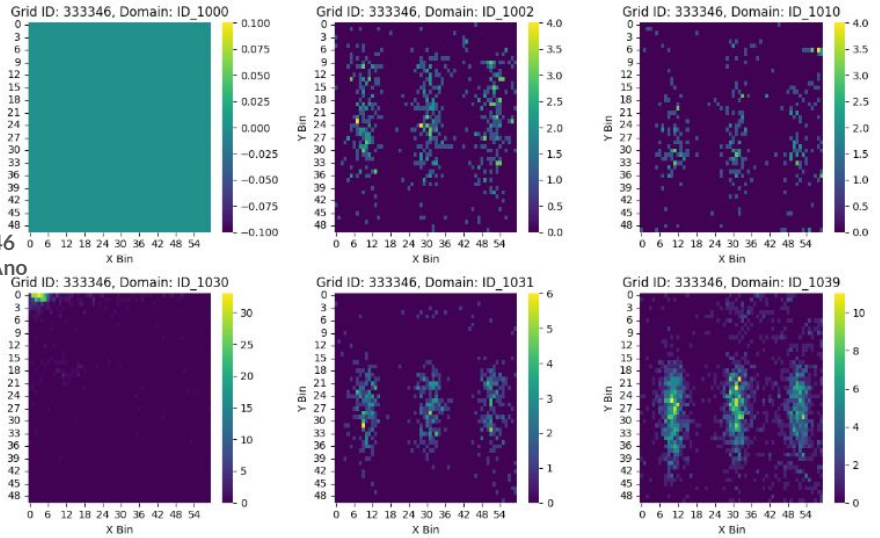
  %: 10.526315789473683 %

  Count: 6

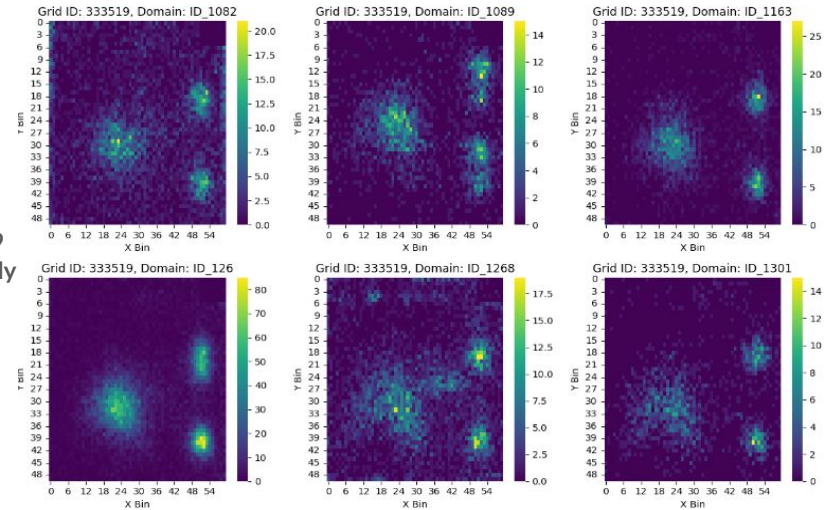
333346 Anomaly



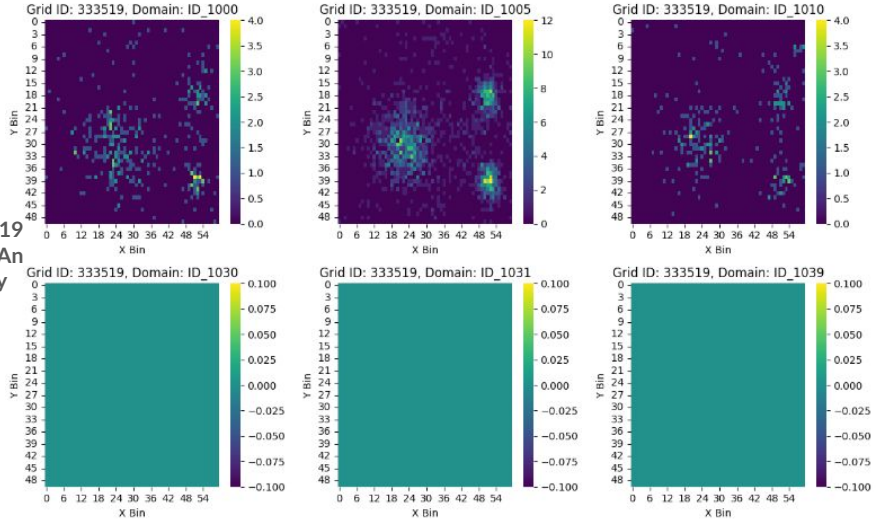
333346 Non-Anomaly



333519 Anomaly

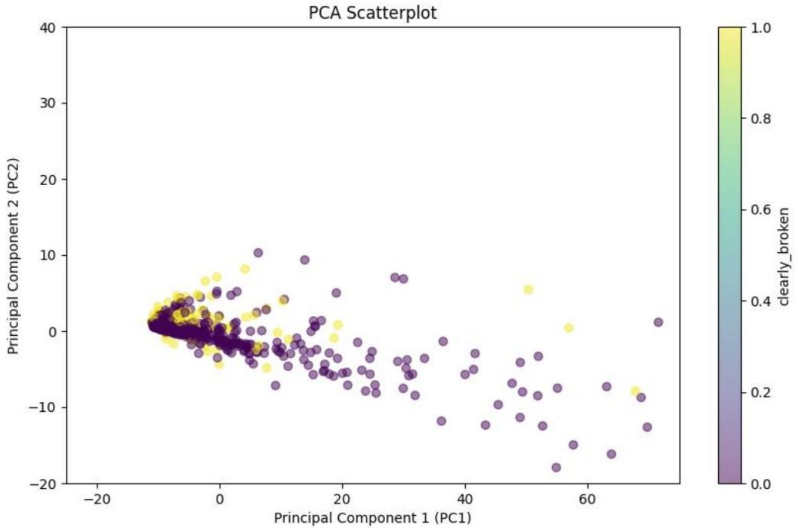


333519 Non-Anomaly

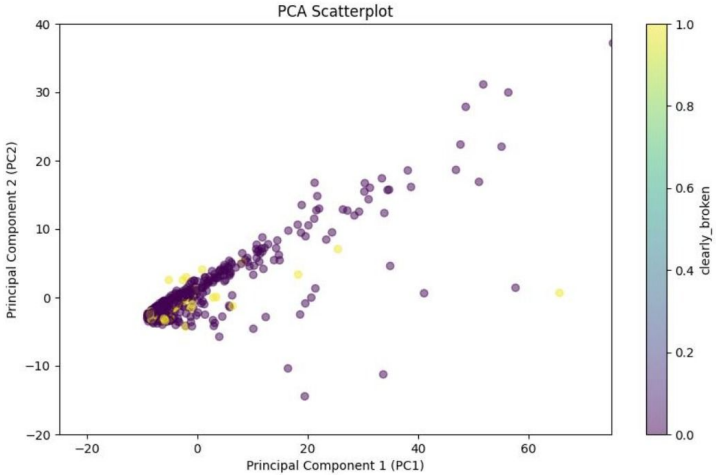


Unsupervised learning on heatmaps dataset

PCA on Heatmap data (scatterplot)



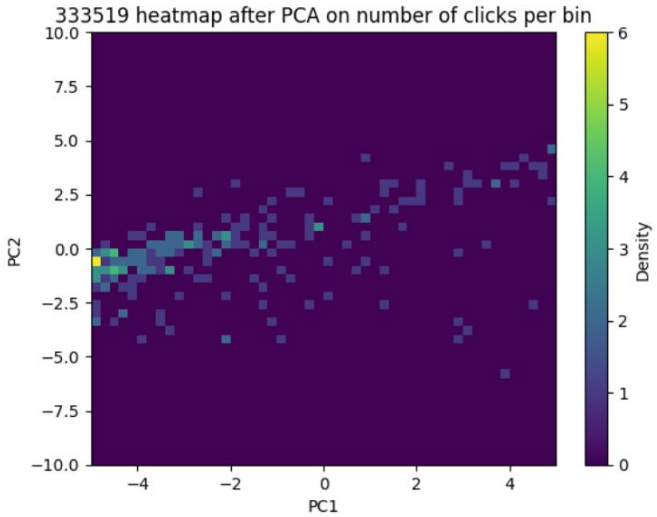
grid_id = 333346

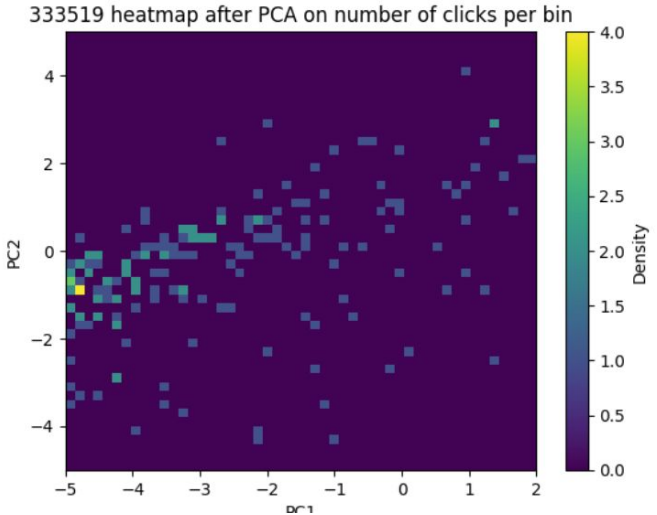


grid_id = 333519

First attempt at PCA: Unable to cluster after conducting PCA on 3000 dimension vector

PCA on Heatmap data (heatmap)





Visualizing the plot as a heatmap: high concentration of points of both classes around 0,0

Other updates

- PCA explained variance ratio: 0.528, 0.051. More than half of the variance is captured by the first PC vector, and less than 5% in the second, less than 4% in the third...
- K means on heatmap data (curse of dimensionality) classifies all points as class 0 and one as class 1

Unsupervised learning on heatmaps dataset

Attempted

- PCA from dim = 3000 to 2
- K-means for k = 2

Possible ideas

- Feature engineering on the heatmaps dataset (resnet-50) to detect
- Grid search on hyperparameters

PCA explained variance ratio and summary

```
1  pca.explained_variance_ratio_
```

[2] ✓ 0.0s

```
array([0.52799197, 0.051471  ])
```

K = 2

```
array([0.52799197, 0.051471  , 0.03832903, 0.02343564, 0.01968418,
       0.01554966, 0.0131    , 0.00931461, 0.0084494 , 0.00832934])
```

K = 10

```
1  print(result["PC1"].describe())
```

[43] ✓ 0.0s

```
count    8.720000e+02
mean    -7.822489e-16
std      4.427892e+01
min     -9.100912e+00
25%     -7.972913e+00
50%     -6.471793e+00
75%     -3.143834e+00
max      1.118921e+03
Name: PC1, dtype: float64
```

```
1  print(result["PC2"].describe())
```

[44] ✓ 0.0s

```
count    8.720000e+02
mean    -3.259370e-17
std      1.312584e+01
min     -1.858551e+02
25%     -2.326660e+00
50%     -1.791962e+00
75%     -4.578672e-01
max      2.870244e+02
Name: PC2, dtype: float64
```

K means on heatmap data

```
1  grid_id
```

76] ✓ 0.0s

```
333519
```

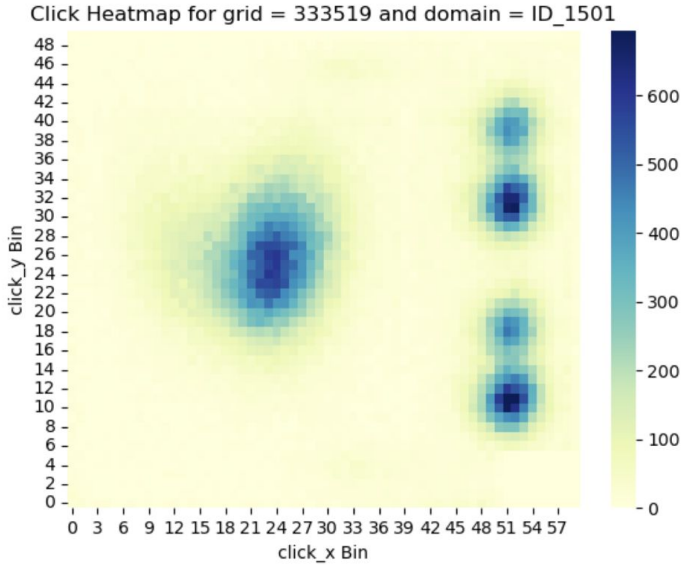
```
1  from sklearn.cluster import KMeans
2  import numpy as np
3
4  # X = np.array([[1, 2], [1, 4], [1, 0],
5  #               [10, 2], [10, 4], [10, 0]])
6
7
8  kmeans = KMeans(n_clusters=2, random_state=0, n_init="auto").fit(result[col_names])
9  print(sum(kmeans.labels_))
10 # kmeans.predict([[0, 0], [12, 3]])
11 # kmeans.cluster_centers_
```

74] ✓ 0.0s

```
1
```

K-means Clustering on metrics dataset

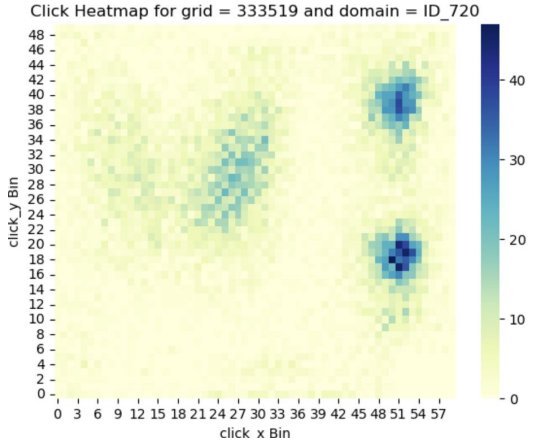
- A lot of missing values: Replace null values with the mean of each column → not reliable
- Normalized data
- Applied k-means within grid_id 333519
- with k=3 → 'ID_1501' is the only one in cluster 1, and its heatmap looks pretty normal

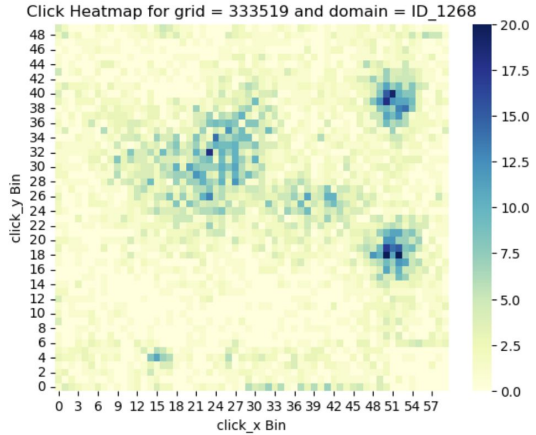


0 2879
2 61
1 1
Name: cluster, dtype: int64

- k=4 → some grid_id and domain in metrics data don't exist in heatmap data, e.g. 'ID_1413'

  1 1484
  0 1395
  3 61
  2 1
  Name: cluster, dtype: int64
- No obvious anomaly in some heatmaps in cluster 3
- Percentage of already identified broken domains in minority clusters: 24.19%





K-NN

	grid_id	domain	anomaly_score	is_anomaly
0	333346	ID_1	22.896447	False
1	333346	ID_10	159.845436	True
2	333346	ID_1002	23.771816	False
3	333346	ID_1005	62.837153	False
4	333346	ID_1010	17.883612	False
...	...	...	...	...
1728	333519	ID_989	29.311979	False
1729	333519	ID_990	17.780857	False
1730	333519	ID_995	15.671548	False
1731	333519	ID_996	44.081592	False
1732	333519	ID_999	21.029098	False

1733 rows × 4 columns

K-NN

- K=5, Threshold = 95 percentile

Results:

grid_id	anomaly_count	total_domains	anomaly_percentage
333346	56	861	6.504065
333519	31	872	3.555046

Check overlaps with the broken domain got by Martin:

- 2 Overlap for grid_id 333519: {'ID_2076', 'ID_398'}
- Overlap percentage: 3.508771929824561%
- 14 Overlap for grid_id 333519: {'ID_489', 'ID_2569', 'ID_1165', 'ID_3180', 'ID_1568', 'ID_10', 'ID_2836', 'ID_1448', 'ID_1062', 'ID_2883', 'ID_2339', 'ID_3382', 'ID_2076', 'ID_2609'}
- Overlap percentage: 12.389380530973451%

Next steps:

- Plot the heatmap to check detection on anomaly
- Adjust the K and threshold values to find the proper value

Question about Metric dataset:

- Were the datasets "metric" and "heatmap" collected over different periods of time?
  - The banners might not be broken at different time periods? And how can we check the efficacy of any clustering method?
  - If they were collected in the same period, can we assume that a broken heatmap <-> corresponding metric data should also anomalous?
  - Have many more domains in Metric data.

Next steps:

- In depth understand performance of each method.
- Implement fancier method → instead of features being explicit bins/clicks, create more meaningful feature vectors:
  - Use pretrained ResNet/ViT/... and run clustering on feature vectors of heatmaps
  - Train Auto-Encoder on arbitrary synthetic clusterings (ask AE to recreate original image with discriminative loss).
  - Train/Fine-tune ResNet/ViT/... on synthetic data to count number of clusters → might lead to interesting feature vectors.
- Combine both datasets in a meaningful way (see next slide), and engineer combined features.
- Further research into clustering methods and dimensional reductions.
- Explore ensembles of basic methods to boost efficacy.

Additional Questions

-