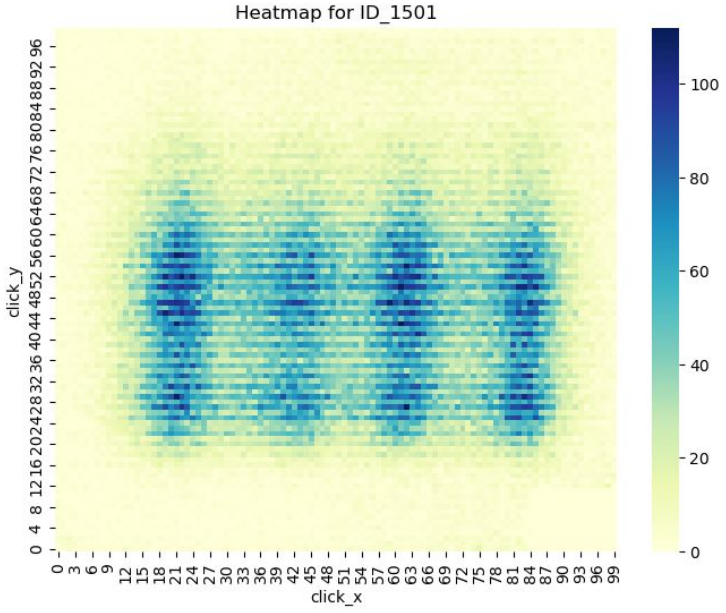
Capstone Project: Heatmap Anomaly Detection

Week 11 Progress Report

This week:

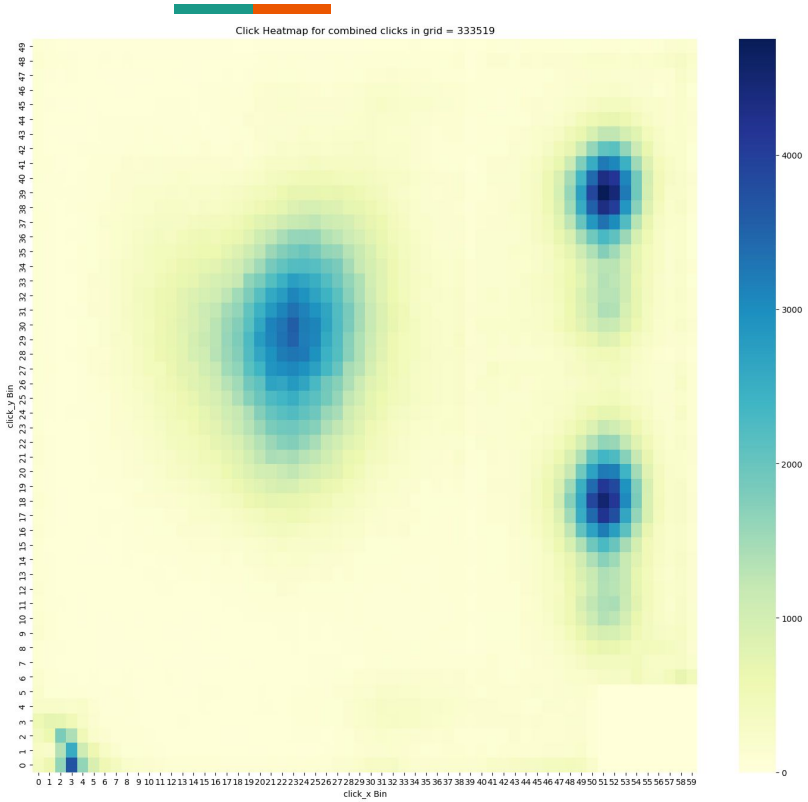
1. Old/New Statistical Approach
   a. LRT
   b. Click-clustering
2. More results from new data
   a. ResNet/ViT
   b. Clustering
3. “Clustering” broken banners
   a. Structure of broken banners → generate synthetic data
   b. Measure distance from characteristic broken banners

Questions



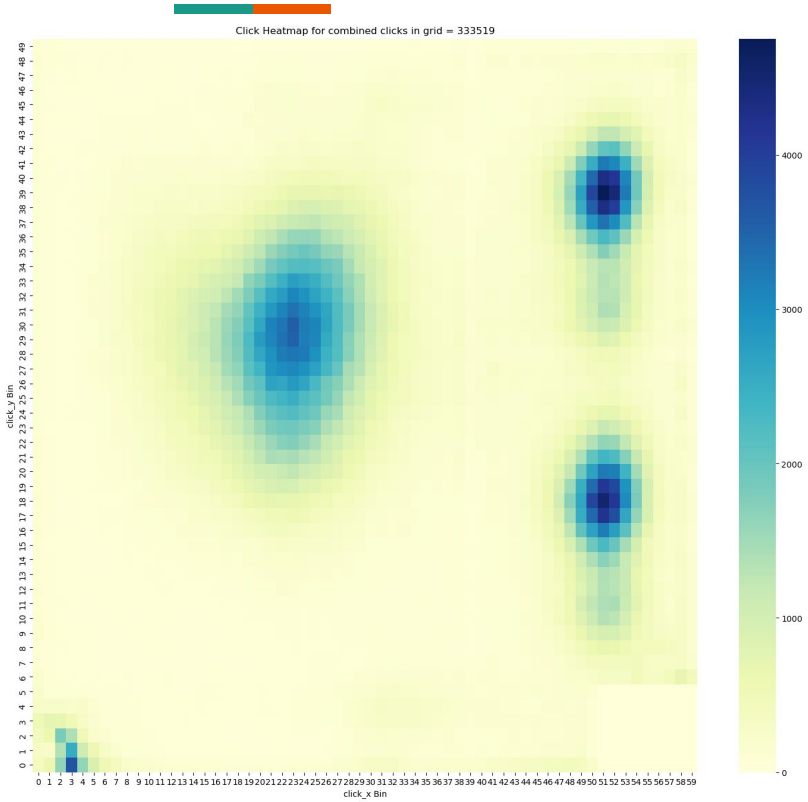
- New dataset:
  - Overlap between metrics and heatmap data quite limited.
  - Suspect that “long” clusters are actually two products in the middle?

Old/New Statistical approach (wip):



- Cumulative clicks → empirical distribution f
- New banner → drawn from distribution
- **Draw probability:**
  - Can compute P[New banner|emp distribution]
  - Normalize: p = P[NB|ED]/P[Avg B|ED]
  - If p < threshold: broken.
    - ~30% identified as broken
- **Chi-squared:**
  - Chi-squared test:
  - Is the underlying distribution the same between ED and NB?
    - ~30% identified as broken
- **LRT:**
  - Estimate “bad banner” distribution
  - Compute p = P[NB|ED]/P[Avg B|BB]
    - Likelihood-ratio-test (to do).
- **Upshot:**
  - Lightweight and can be easily implemented on data-stream.
- **To do:**
  - Different resolutions
  - Upsample/downsample
  - “pseudo-distance” between good and bad banners (e.g. KL-divergence)

Old/New Statistical approach (wip):

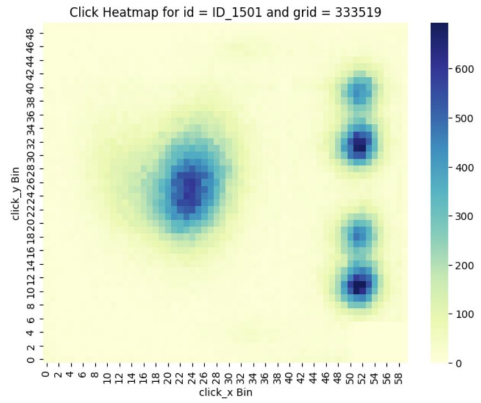


- Cumulative clicks → empirical distribution f
- New banner → drawn from distribution
- **Draw probability:**
  - Can compute P[New banner|emp distribution]
  - Normalize: p = P[NB|ED]/P[Avg B|ED]
  - If p < threshold: broken.
    - ~30% identified as broken
- **Chi-squared:**
  - Chi-squared test:
  - Is the underlying distribution the same between ED and NB?
    - ~30% identified as broken
- **LRT:**
  - Estimate “bad banner” distribution
  - Compute p = P[NB|ED]/P[Avg B|BB]
    - Likelihood-ratio-test (to do).
- **Upshot:**
  - Lightweight and can be easily implemented on data-stream.
- **To do:**
  - Different resolutions
  - Upsample/downsample
  - “pseudo-distance” between good and bad banners (e.g. KL-divergence)

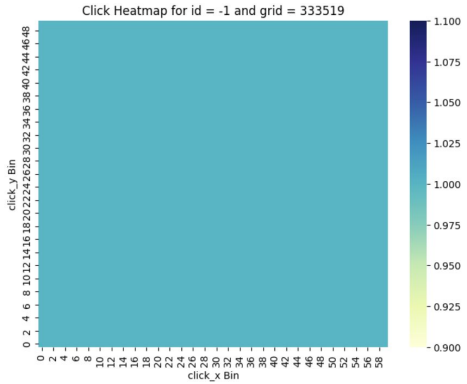
LRT method

Intuition: classify a banner as broken (1) or working (0) by evaluating its similarity to a "representative banner"

Representative good banner



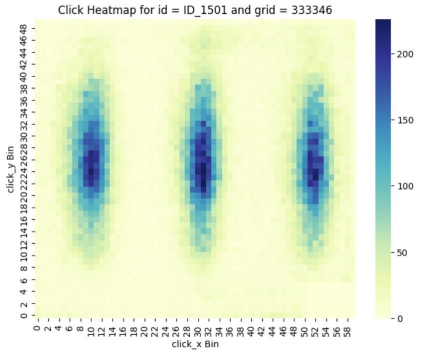
Representative broken banner



Good representative banners were chosen by filtering based on the percentage of successful landed clicks and picking the banner with the largest number of clicks, whereas bad representative banners are modelled with the uniform distribution

LRT method

Step 1: Choose the representative banners from the first/original heatmap dataset
Step 2: Perform LRT on all the banners in the first heatmap dataset
Grid search by varying the size of the upsampling, the probability threshold for classification as good or broken banner

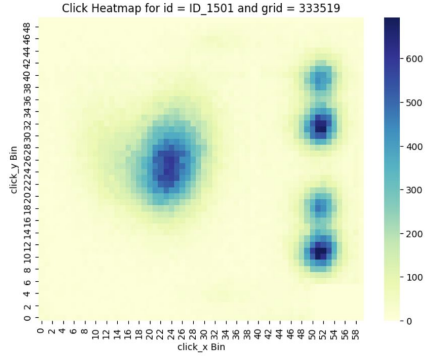


333346	True 0	True 1
Pred 0	706	2
Pred 1	42	111

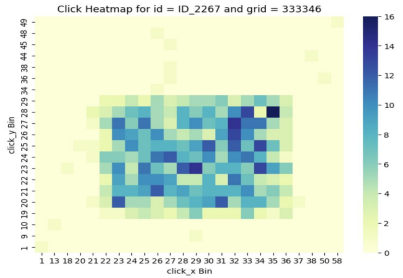
Accuracy: 0.9489
Precision: 0.7255
Recall: 0.9823
F1-score: 0.83459

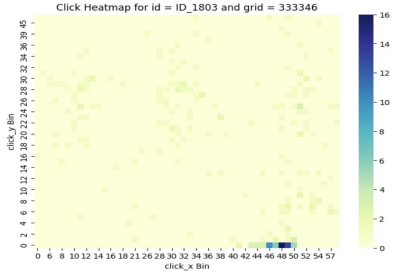
333519	True 0	True 1
Pred 0	795	1
Pred 1	20	56

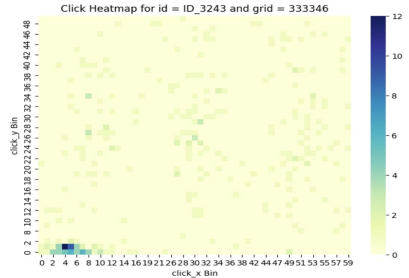
Accuracy: 0.9759
Precision: 0.7368
Recall: 0.9825
F1-score: 0.8421

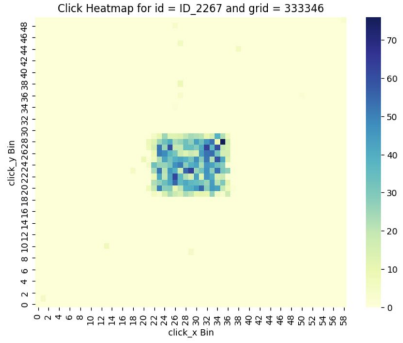


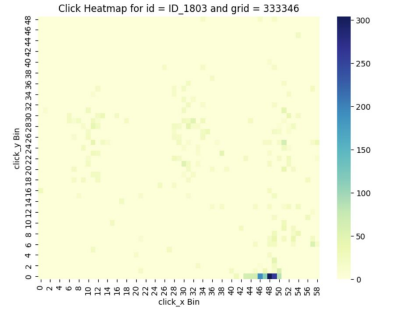
LRT method: false negatives (broken banners misclassified as working)

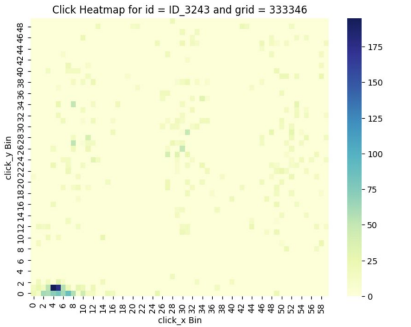








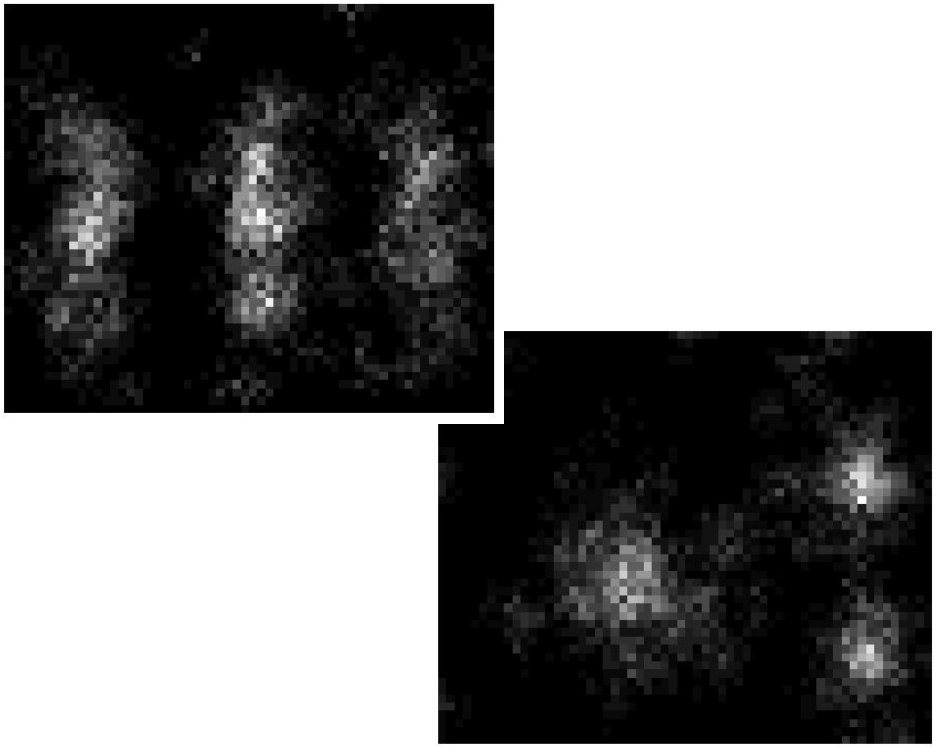




LRT method: Next steps

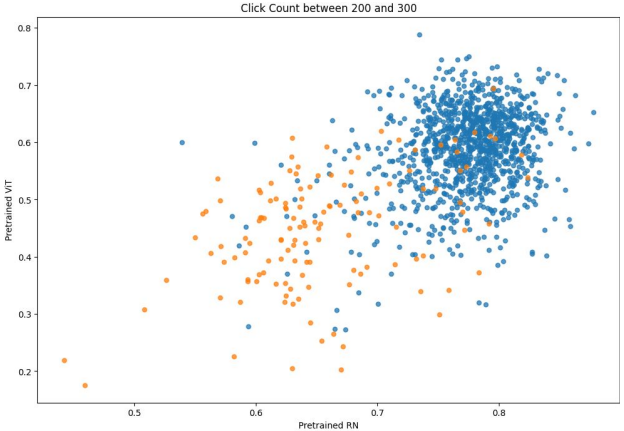
- Finetuning transfer learning across grid types
- Further analysis on misclassified cases, representative heatmaps

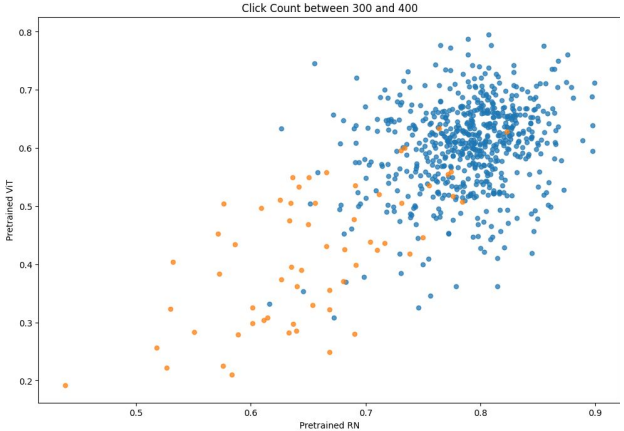
Pretrained ViT/ResNet (recap):

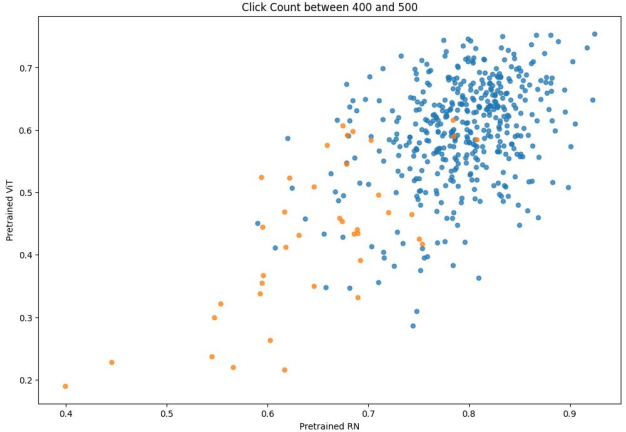


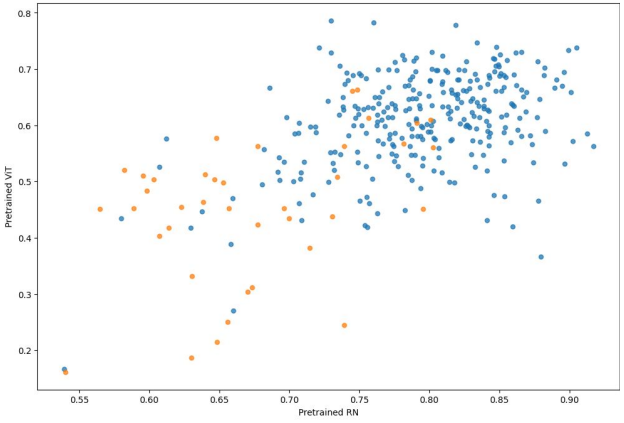
- Feed (transformed and binned) heatmaps into pre-trained ViT/ResNeT.
  - google/ViT: Transformer-based architecture, 14M images (224x224), 21k classes
  - Microsoft/ResNet-1k: trained on ImageNet-1k (224x224), 1k classes.
- Extract features (before classification head)
  - ViT → 151296 dim'l feature vector
  - ResNet → 2048 dim'l feature vector
- Play with upsampling (bootstrapping + noise)
- PCA and other dim'l reduction techniques
  - Apply clustering methods

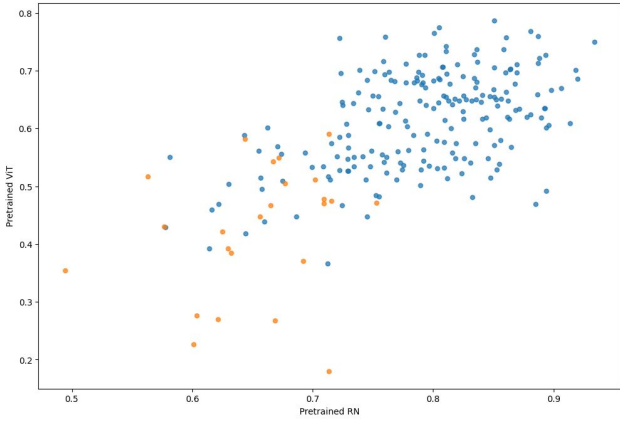
Pretrained ViT/ResNet (facet by click count):

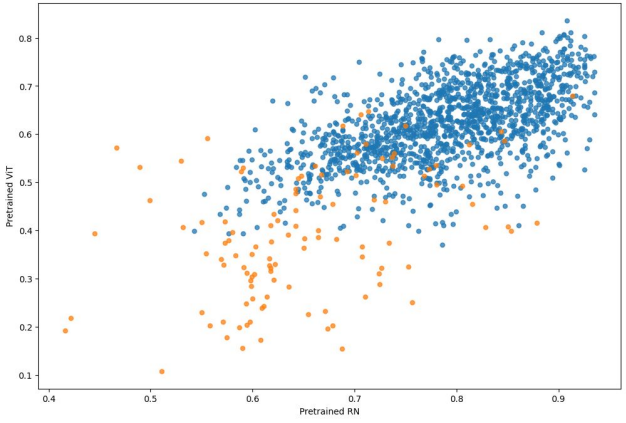




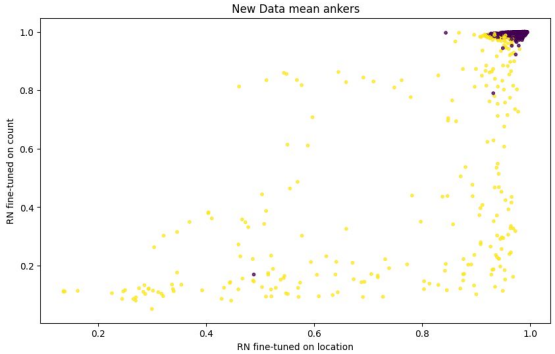


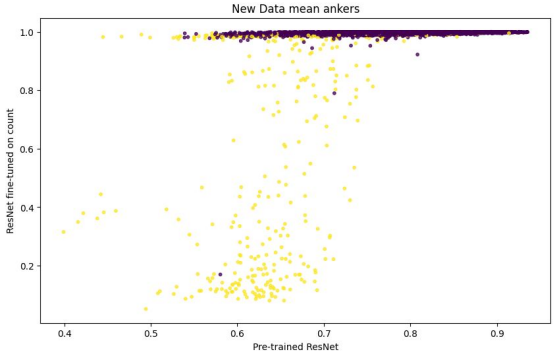


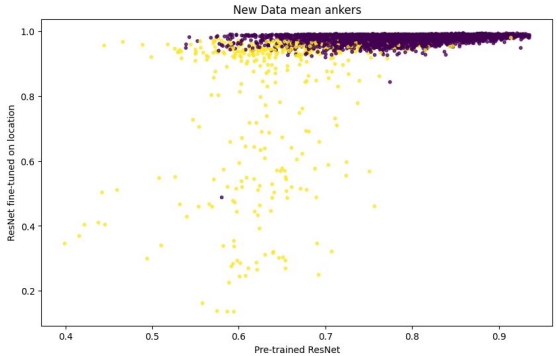


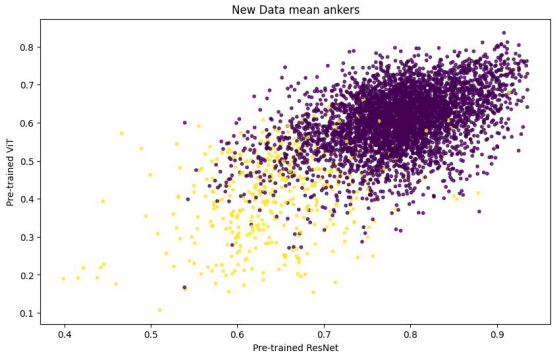


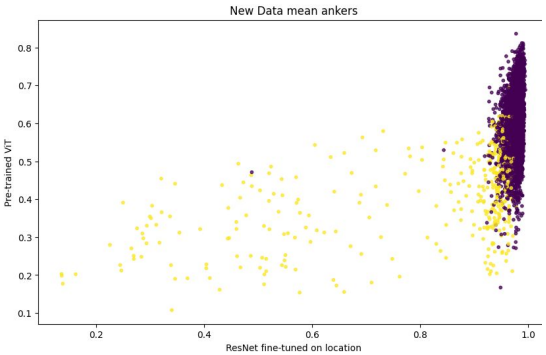
All data cosine-simularities

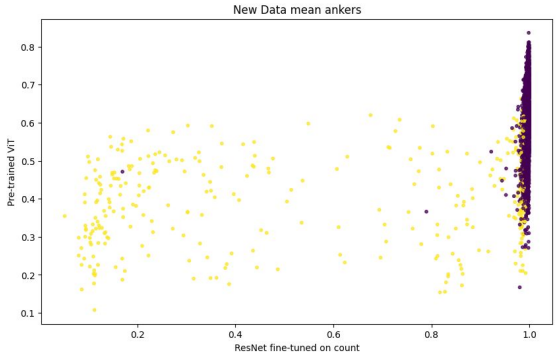




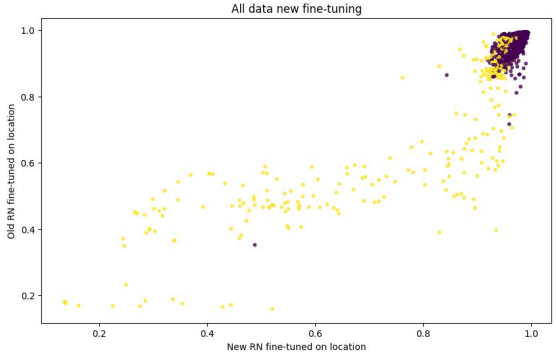


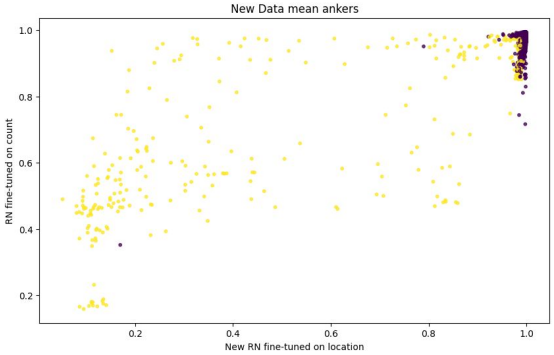


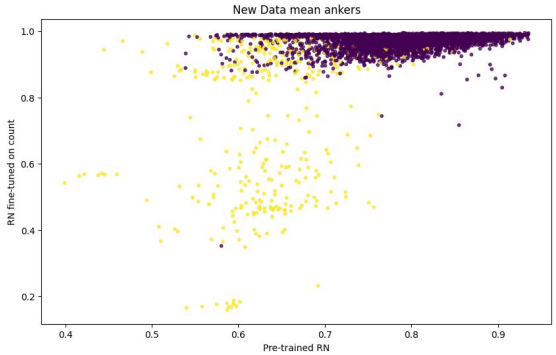


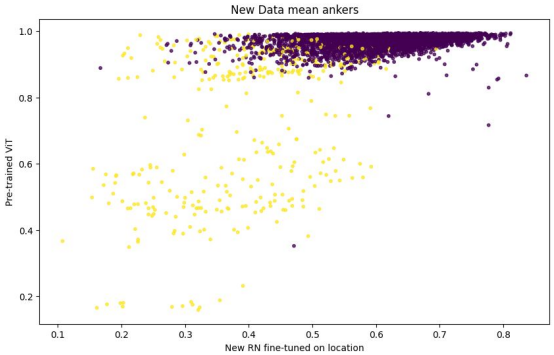


All data: Better synthetic data



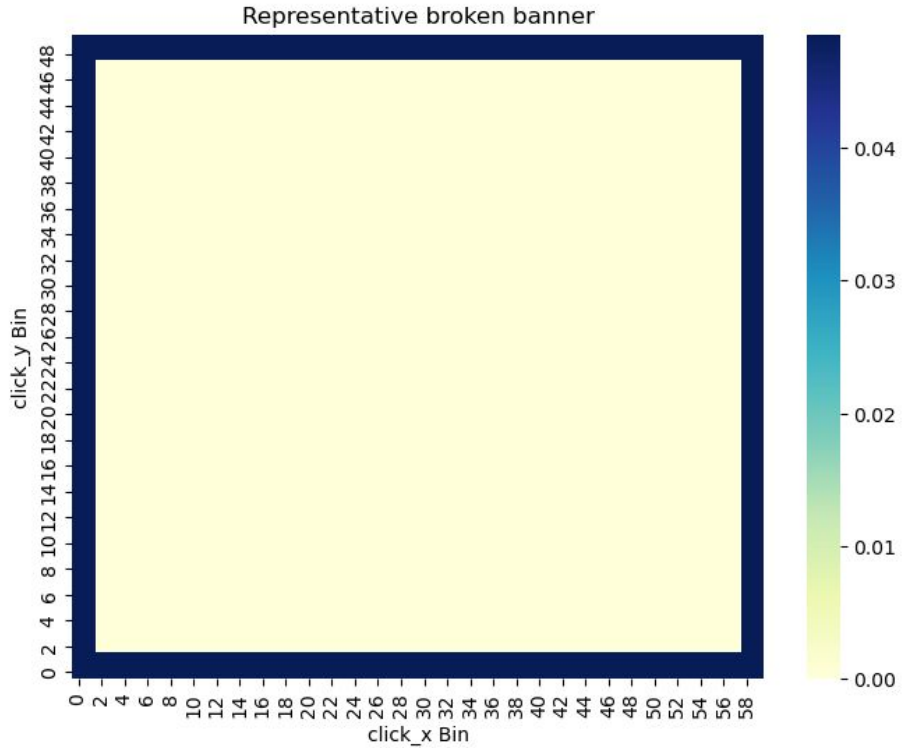






- New method for predicting locations of clusters.
  - Cleaner synthetic data
  - No overlaps
  - More noise

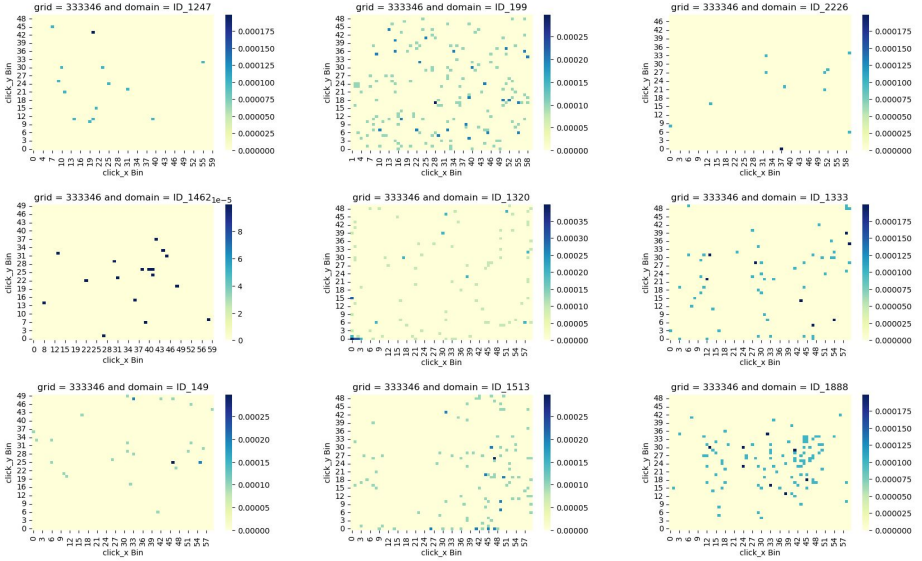
Toy example:



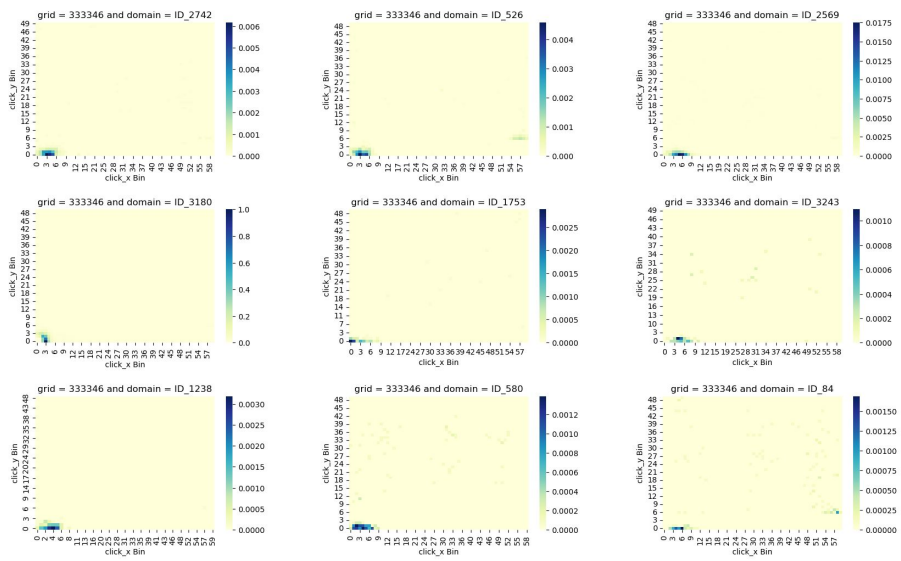
- Compute Euclidean distance (=cosine-similarity) between heatmaps and left banner:
  - Run over threshold distance:
  - New data:
    - **Threshold: 0.11**
    - ((TN,FN),(FP,TP)) = (2768,73),(36,153)
    - F1-score: 0.73735
  - Old data:
    - **Threshold: 0.11**
    - ((TN,FN),(FP,TP)) = (1493,70),(28,142)
    - F1-score: 0.74346

211 cb in old data
227 cb in new data

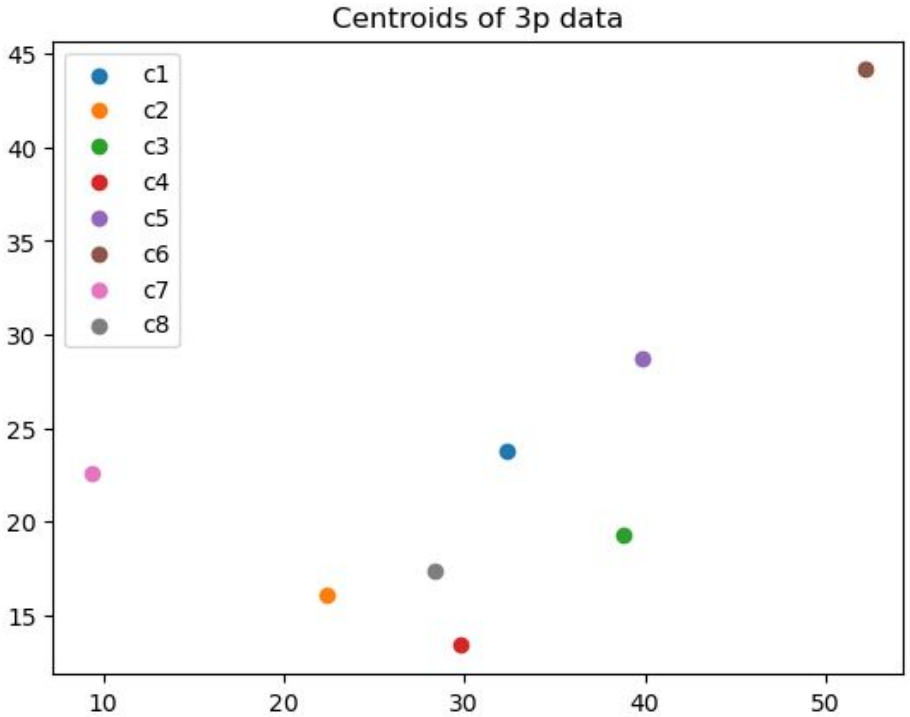
Clustering by broken heatmaps centroids



Cluster 1

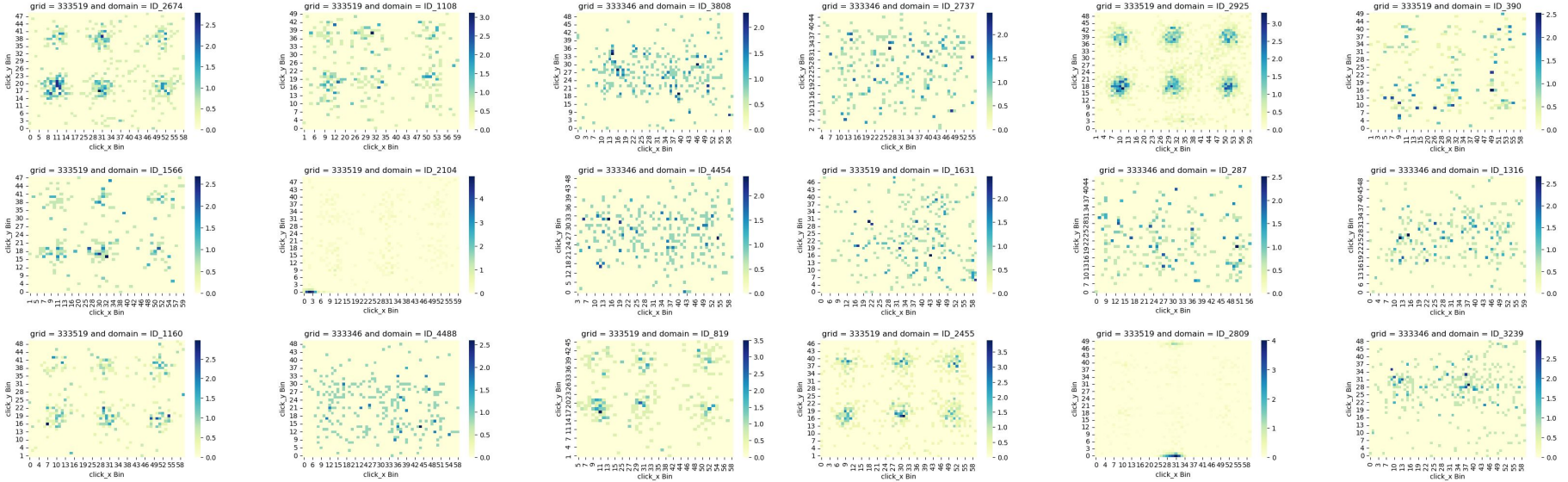


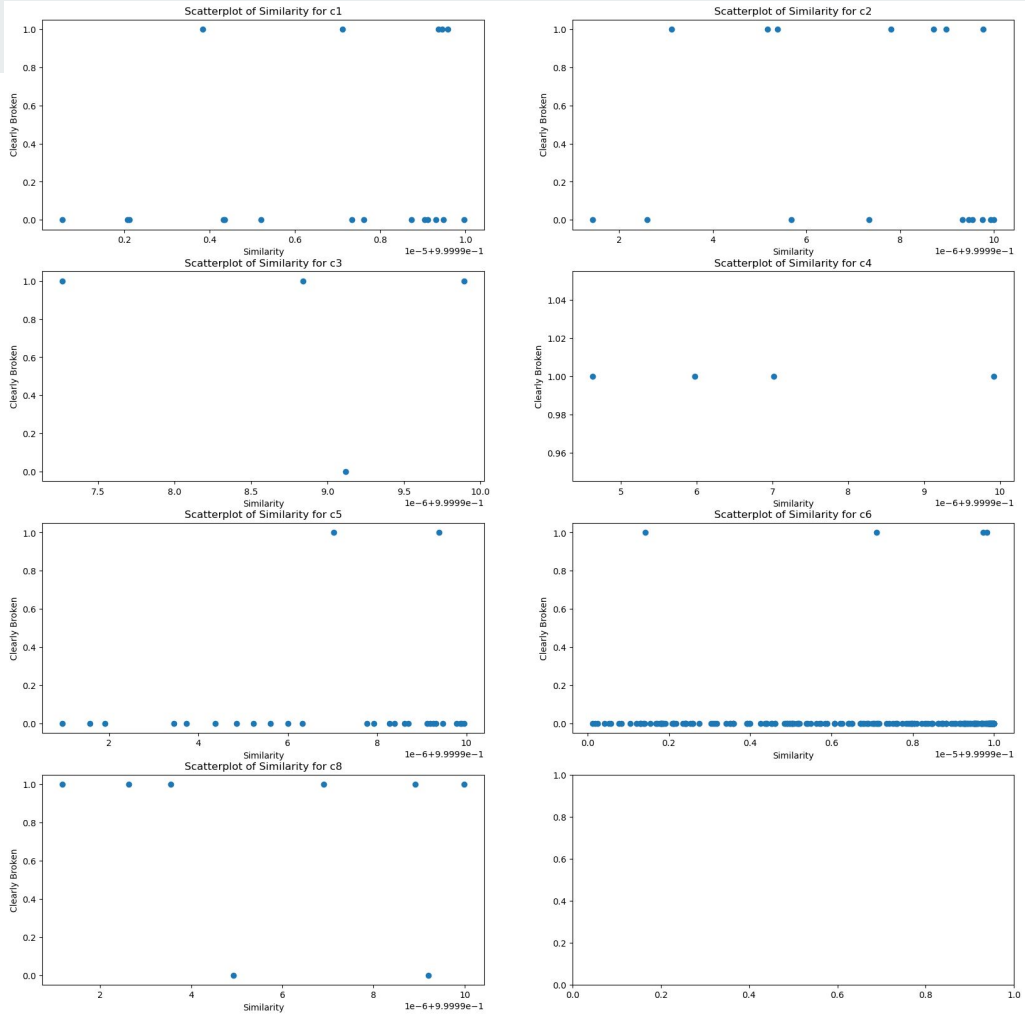
Cluster 2



c1: all over the place
c2: left bottom corner
c3: right bottom corner
c4: bottom middle
c5: right middle
c6: top right corner
c7: all over left side
c8: diagonal

Cosine Similarity (314 identified, 31 cb)





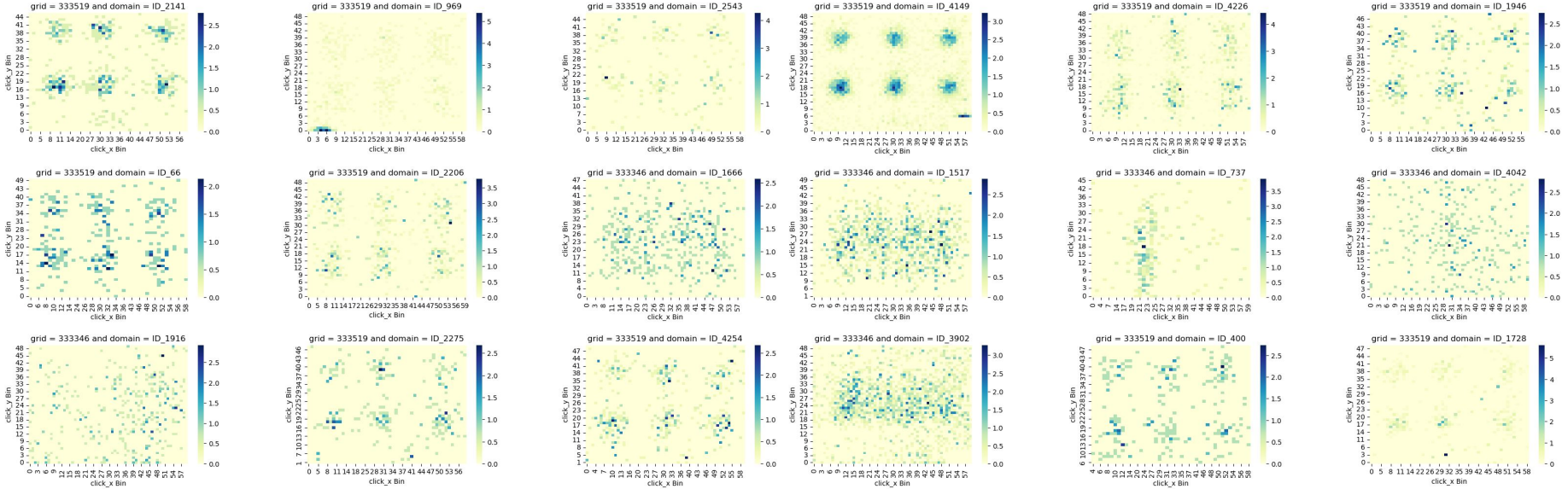
c1: all over the place
c2: left bottom corner
c3: right bottom corner
c4: bottom middle
c5: right middle
c6: top right corner
c7: all over left side
c8: diagonal

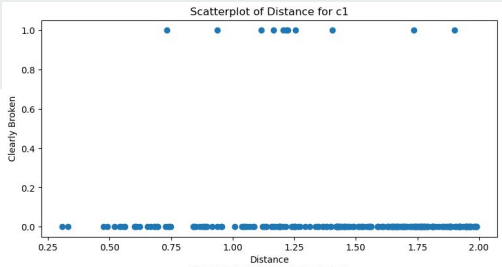
threshold = 0.99999

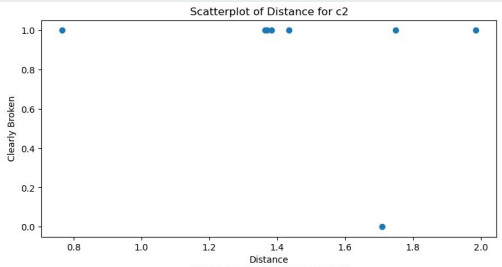
	grid_id	domain	cluster	similarity	clearly_broken
0	333346	ID_1060	c6	0.999992	0
1	333346	ID_1061	c6	1.000000	0
2	333346	ID_1073	c1	0.999998	0
4	333346	ID_1173	c5	0.999998	0
5	333346	ID_121	c1	0.999999	0
...	...	...	...	...	...
338	333519	ID_932	c6	0.999991	0
339	333519	ID_935	c5	0.999991	0
340	333519	ID_956	c1	0.999999	0
342	333519	ID_982	c6	0.999996	0
343	333519	ID_992	c6	1.000000	0

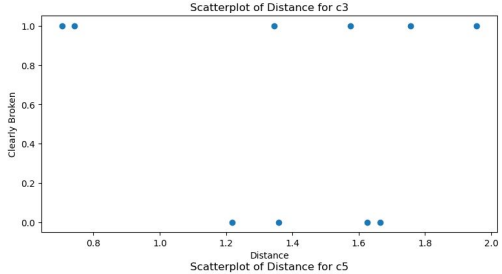
314 rows × 5 columns

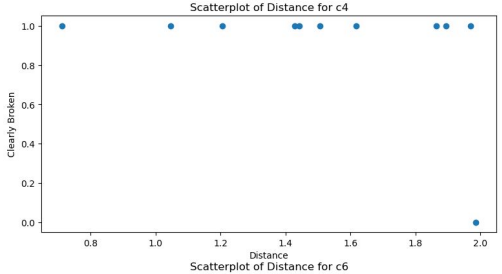
Euclidean Distance (277 identified, 40 cb)

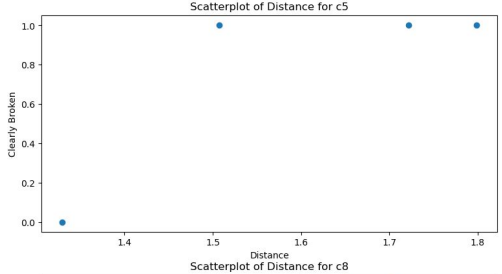


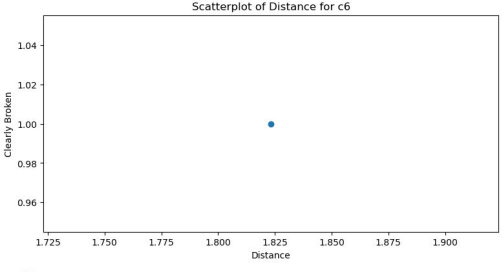


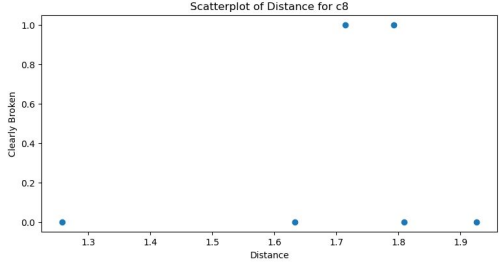


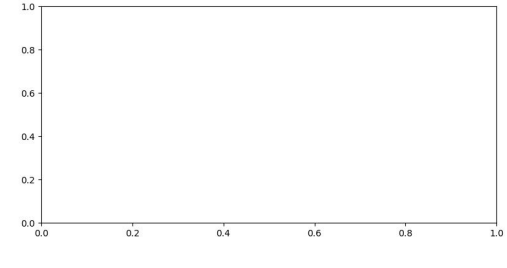












c1: all over the place
c2: left bottom corner
c3: right bottom corner
c4: bottom middle
c5: right middle
c6: top right corner
c7: all over left side
c8: diagonal

threshold = 2

	grid_id	domain	cluster	similarity	clearly_broken
0	333346	ID_1073	c1	0.603229	0
1	333346	ID_1107	c1	1.696355	0
2	333346	ID_1158	c1	1.157189	0
3	333346	ID_121	c1	1.191031	0
4	333346	ID_1232	c1	1.952549	0
...	...	...	...	...	...
276	333519	ID_905	c1	1.777646	0
277	333519	ID_949	c1	0.622101	0
278	333519	ID_963	c1	1.989187	0
279	333519	ID_969	c2	1.433809	1
280	333519	ID_996	c1	1.238415	0

277 rows × 5 columns

Test on New Data

- Apply our trained model to full new data

	On new data	On previous data(test set)
SVM:	F1 Score: 0.23	F1 Score: 0.25
KNN:	F1 Score: 0.64	F1 Score: 0.83
K-Means:	F1 score: 0.04	F1 score: 0.24
DBScan:	F1 Score: 0.82	F1 Score: 0.72
Isolation Forest:	F1 Score: 0.66	F1 Score: 0.85
Combined Model:	F1 Score: 0.82	F1 Score: 0.87

Next steps:

- Further investigate 0/1-shot performance on new dataset.
- Converge towards actual predictor
- Combine 4 ResNet/ViT features into a meaningful predictor.
  - Are there other synthetic tasks?
  - Combine with metrics features and PCA?
- SimCLR -> get it running
- Measure compute requirements more carefully:
  - Data pre-processing slow (I think can be sped up)
  - Inference times slow-ish → can be parallelized (xCPU-number speedup)

Appendix

LRT method

Grid search: find optimal parameter
Threshold not that important, 1%?
Sd of 0 upsampling of 1k is OK
Change the loop for the threshold, save true positive in a dictionary, go through each domain once

Fix parameters based on three product and see how it works for six product -> how it works on test set

Ideas: upsample or downsample with noise
Look into misclassified cases

Grid search on hyperparameters: size of upsample, probability threshold,