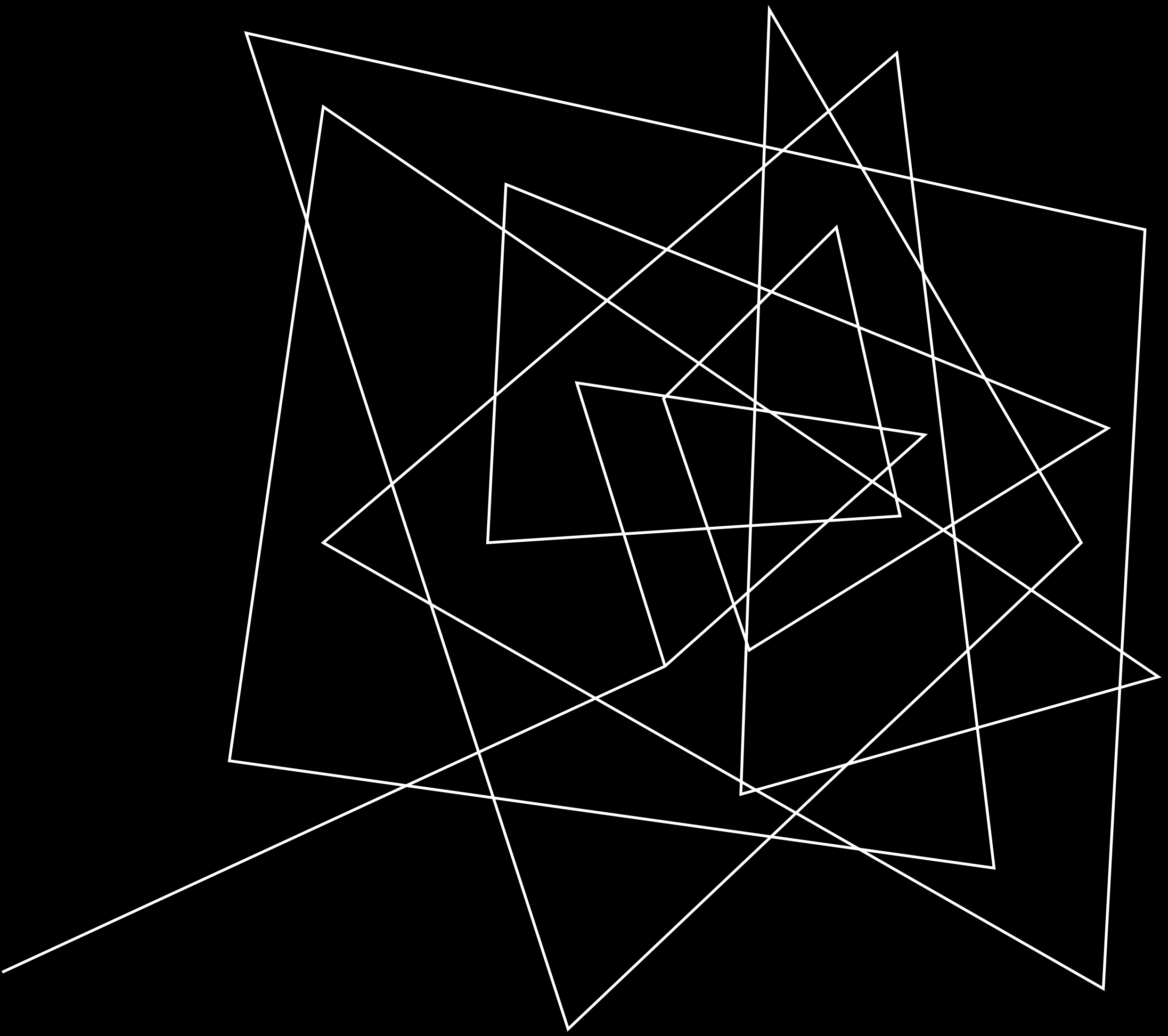
TOPOLOGICAL DATA ANALYSIS

Joshua Sheldon, Justin Barnwell, Michelle Arubi

OUTLINE

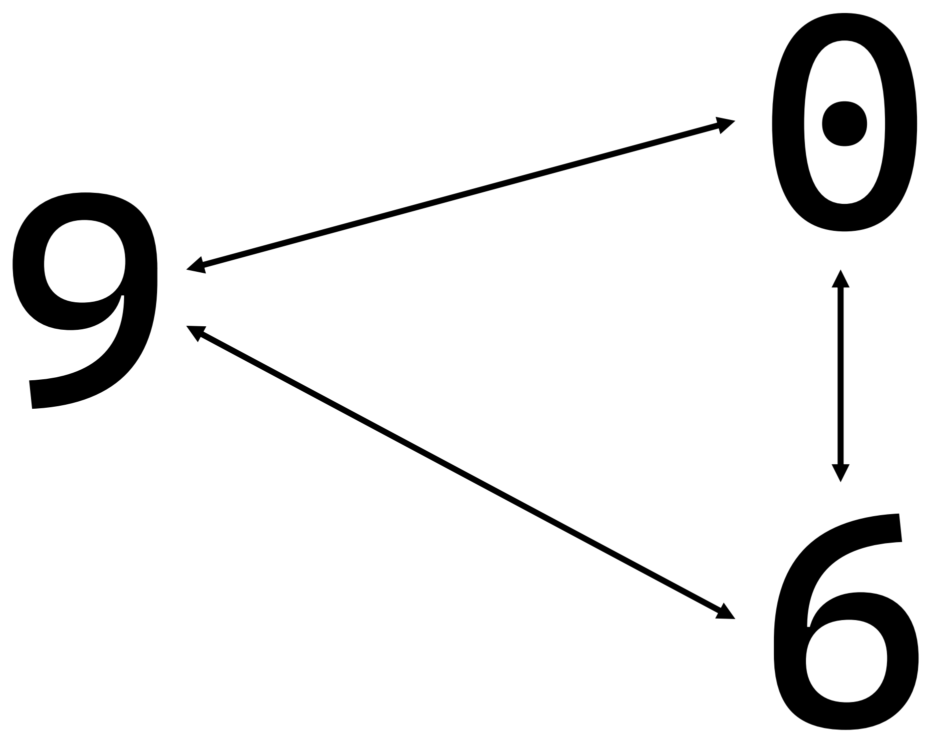
- Intro to TDA
- Data & Objective
- Implementation & Results



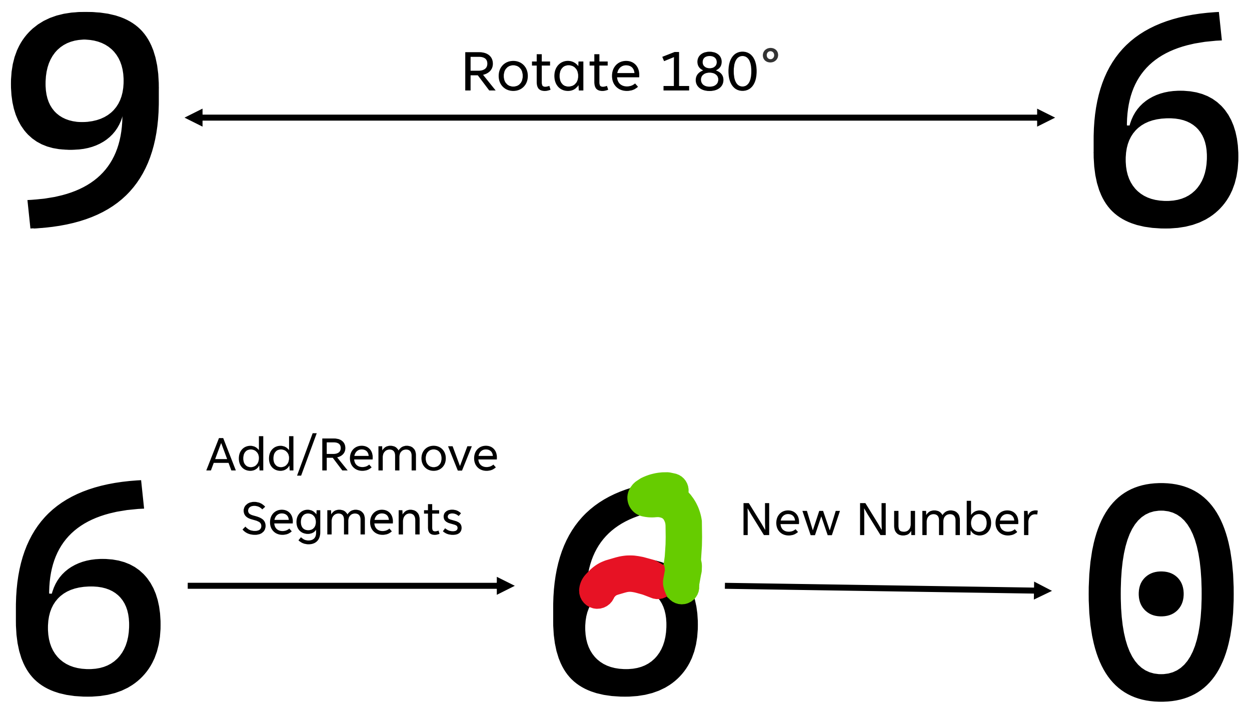
INTRO TO TDA

The science of shapes

SHAPING UP



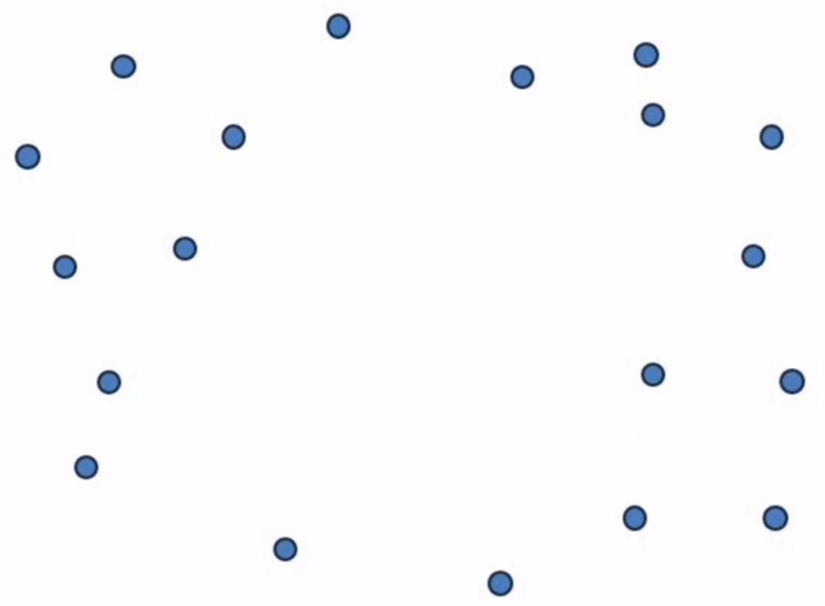
TRANSFORMERS



TOPOLOGICAL DATA ANALYSIS

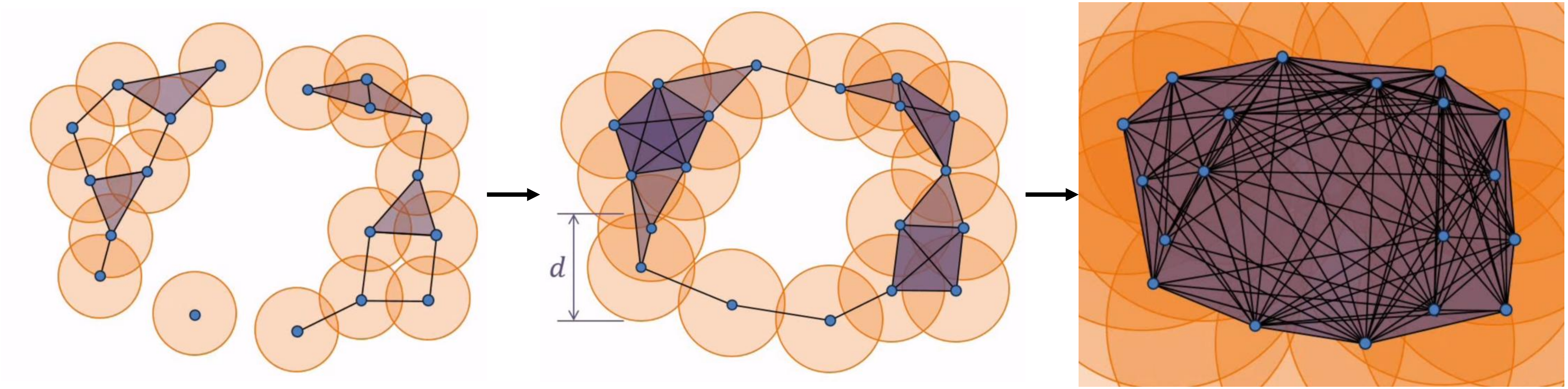
to·pol·o·gy

1. The way in which constituent parts are interrelated or arranged
2. The study of geometric properties and spatial relations unaffected by the continuous change of shape or size of figures



What shape?

PERSISTENT HOMOLOGY



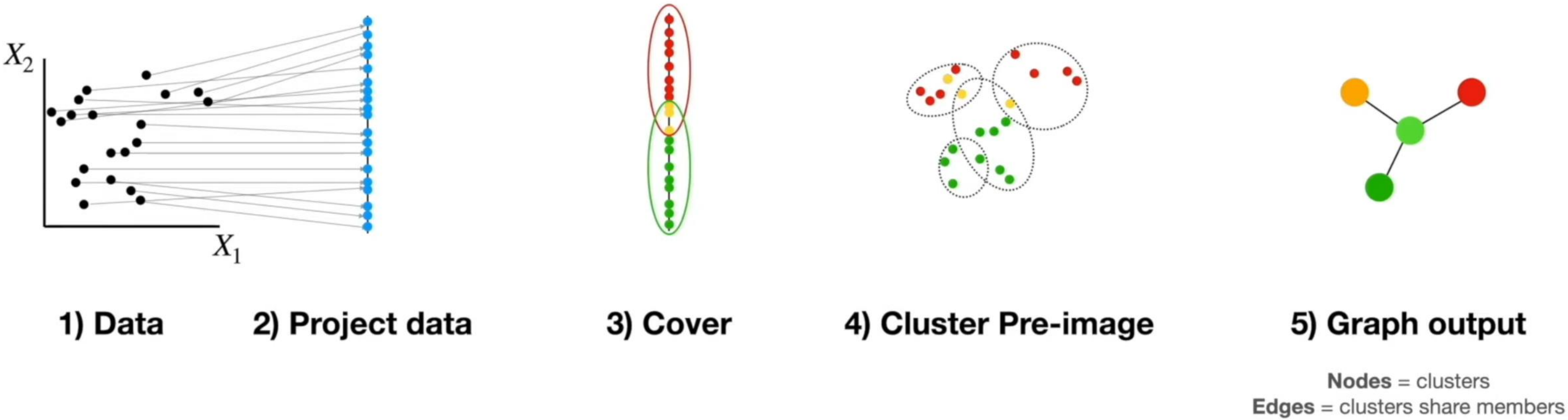
$d = 1$ $d = 2$ $d = 10$

- Middle hole may last from $d = [2, 8]$
- **Persistence** = $d_{end} - d_{start}$
- Higher persistence = feature, lower persistence = noise

NOW WHAT?

- Point clouds = data sets
- Using homology, we acquire **important features** of our data.
- We can reduce the **dimensionality** of data while maintaining **important features**.
- Reduced dimensionality makes **data analysis** possible!

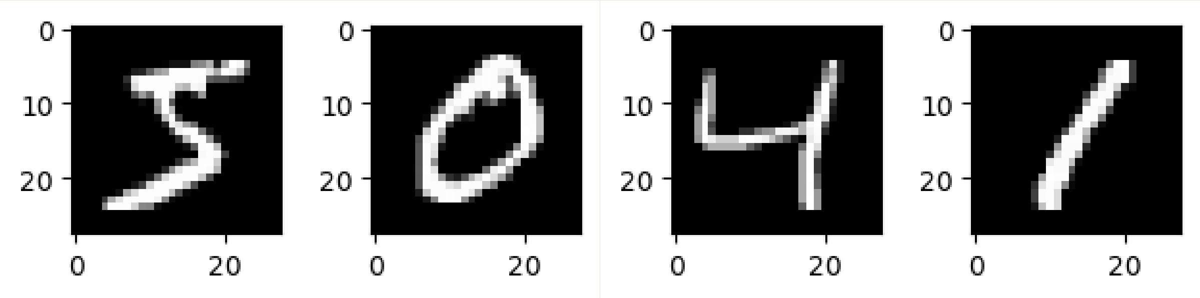
MAPPER ALGORITHM



DATA & OBJECTIVE

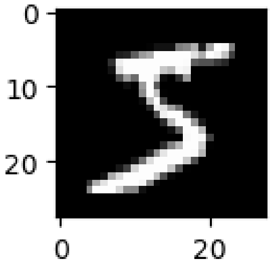
Digit data with detracted dimensionality

MNIST



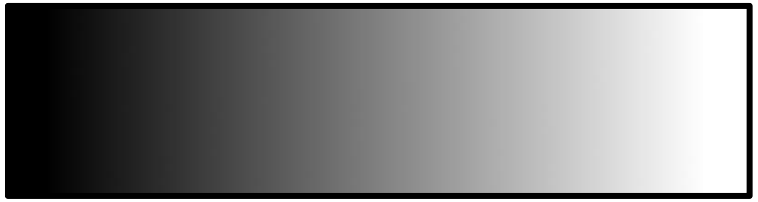
- Library of grasycale handwritten digit images.

MNIST DIGIT EXAMPLE



28x28 matrix of numbers [0, 255] (8 bit unsigned ints)

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136 175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253 225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251  93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  35 241 225 160 108   1   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119  25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253 150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252 253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249 253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253 253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253 250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201  78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
```



0 255

OUR OBJECTIVE

- Select 10 MNIST images for each digit (100 total)
- Analyze them with the Mapper algorithm
- Visualize the relationships between digits

IMPLEMENTATION & RESULTS

You’re in the weeds: beware Pythons

TECH STACK

jupyter

matplotlib

NumPy

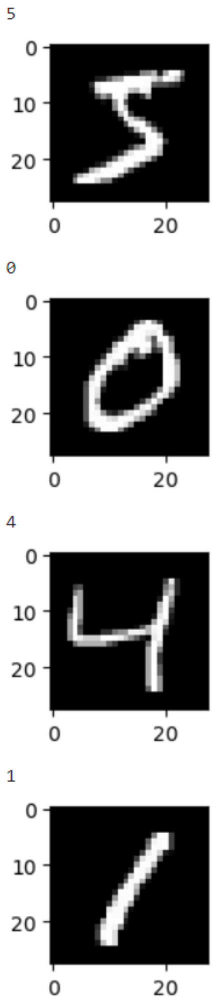
Keras

scikit learn

IMPORTING MNIST

- Keras provides the MNIST dataset through 4 arrays:
  - `x_train : (60000,28,28)`
  - `y_train : (60000,)`
  - `x_test  : (60000,28,28)`
  - `y_test  : (60000,)`
- x = image data
- y = labels

[137]:
```python
# Demonstrating how the x array contains the numbers,
# and the y array contains the labels
for i in range(4):
    print(train_y[i])
    plt.subplot(330 + 1 + i)
    plt.imshow(train_x[i], cmap=plt.get_cmap('gray'))
    plt.show()
```



STORING DIGITS

- Converting 28x28 matrices to 784-dimension vectors
- Three new data structures
  - `data   : array(100,784)`
  - `labels : array(100,)`

SELECTION ALGORITHM

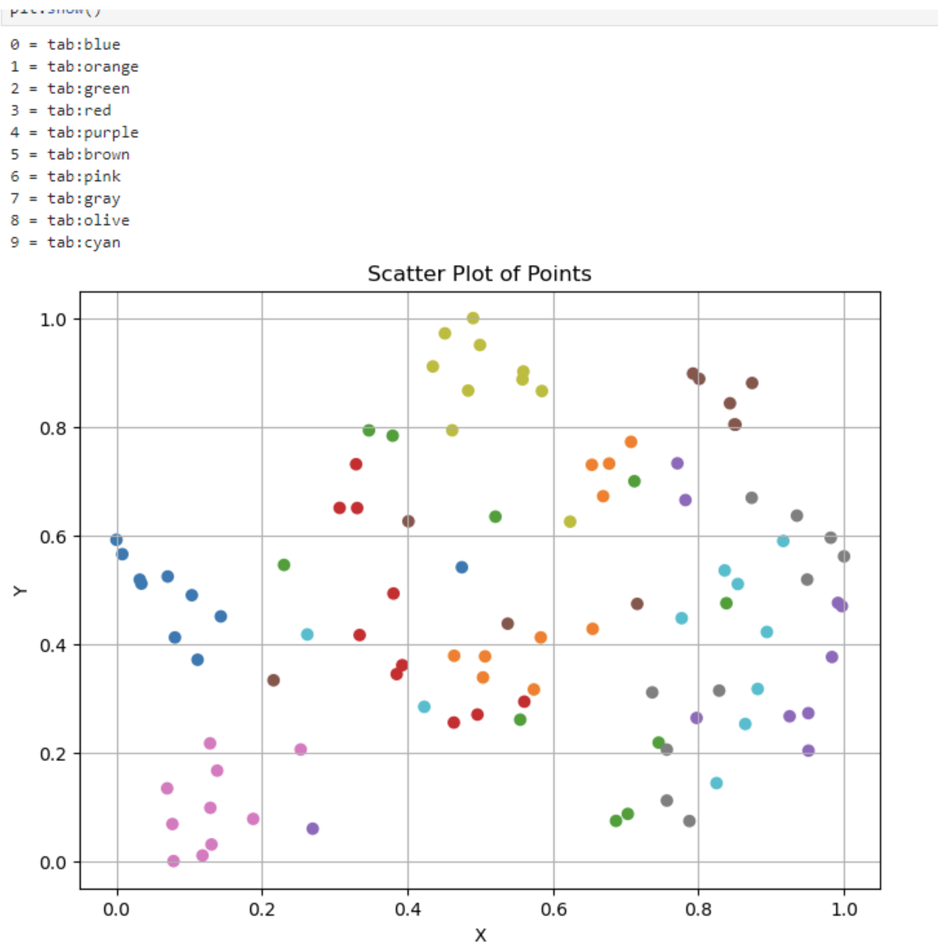
```
# Keep track of how many of each digit we’ve collected
added = new size 10 array, initialized to 0s

# Select digits
old_index = 0 # For traversing through Keras MNIST arrays
new_index = 0 # For traversing through our arrays
while not all(value == 10 for value in added):
    digit = train_y[old_index]

    if added[digit] < 10:
        # Add label in format: <digit> (#<occurrence>)
        labels.append(label)

        # Reduce dimensionality and add to array
        data[new_index] = train_x[old_index].reshape(-1)

        # Increment occurrences of digit and position in data array
        added[digit] += 1
        new_index += 1

    old_index += 1
```

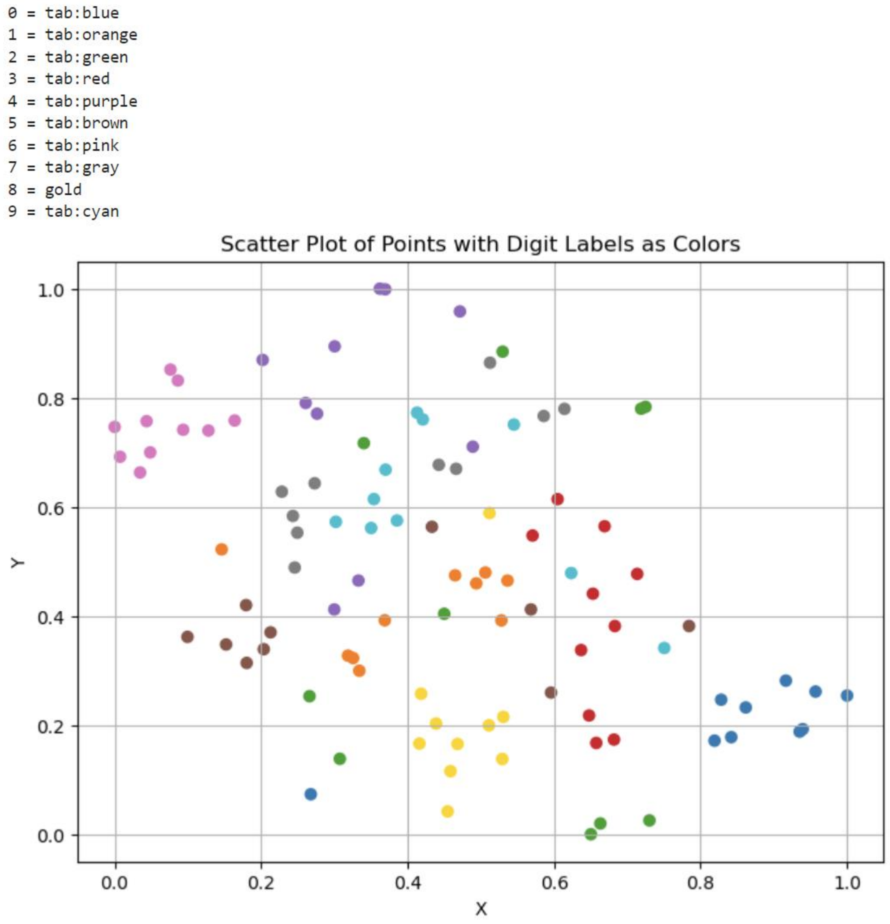
REDUCING DIMENSIONALITY

- Now we apply a series of **projections** to reduce our data from 784 dimensions to 2.
- Example:
  - Isometric Mapping – 784 -> 100
  - UMAP* – 100 -> 2
- Other popular projections: PCA, t-SNE, Feature Scaling
- Each projection may have different strengths/objectives.

PROJECTION AS A LENS

- Pipeline 1
  - Isometric Mapping
  - UMAP
- Pipeline 2
  - MinMax Scaler
  - T-SNE



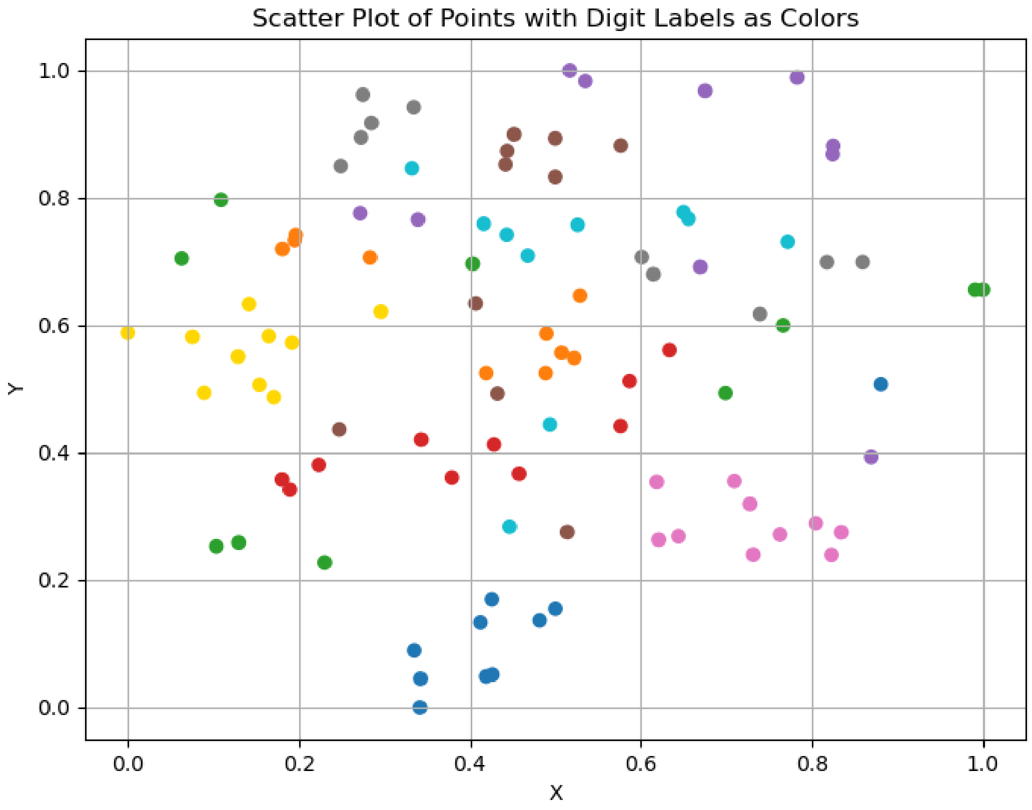


COVER, CLUSTER, AND GRAPH

- Now that our data is in 2 dimensions, we can create a cover, cluster the data, and construct a graph.
- Will refer to these operations as **mapping**.
- We tried two types of mapping: **informed** and **uninformed**.
- The difference between these two is knowledge of the digit that each data point represents.

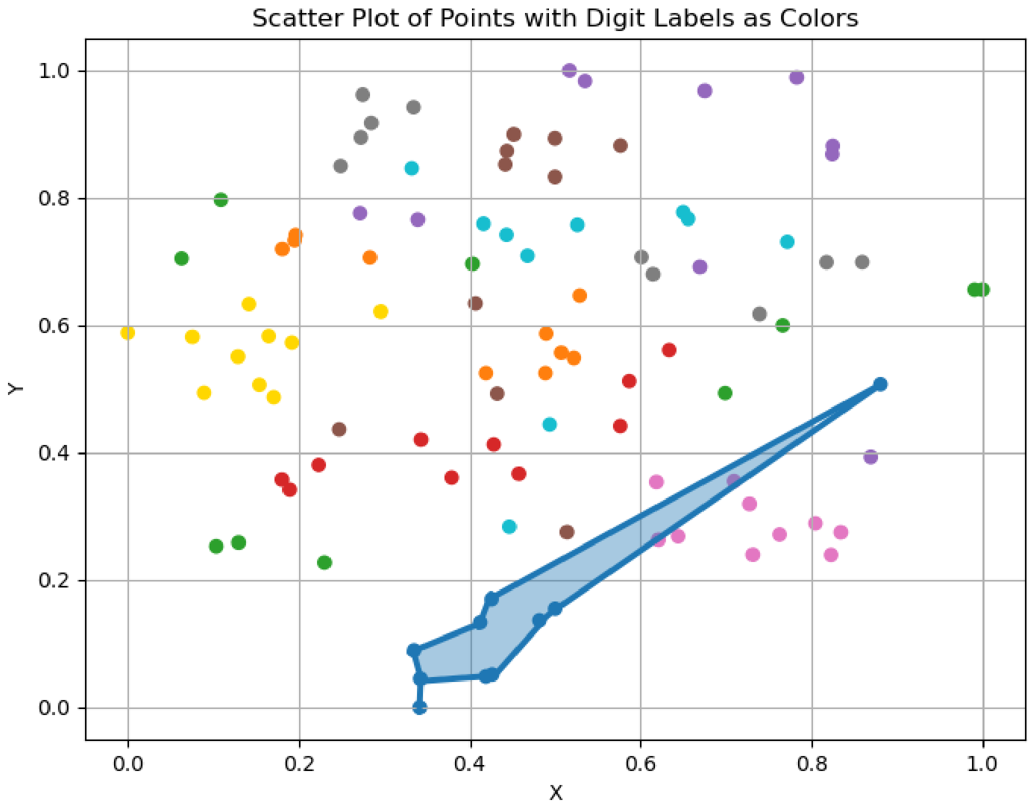
MANUAL MAPPING

Step 1: Create concave hulls around all points of a digit.



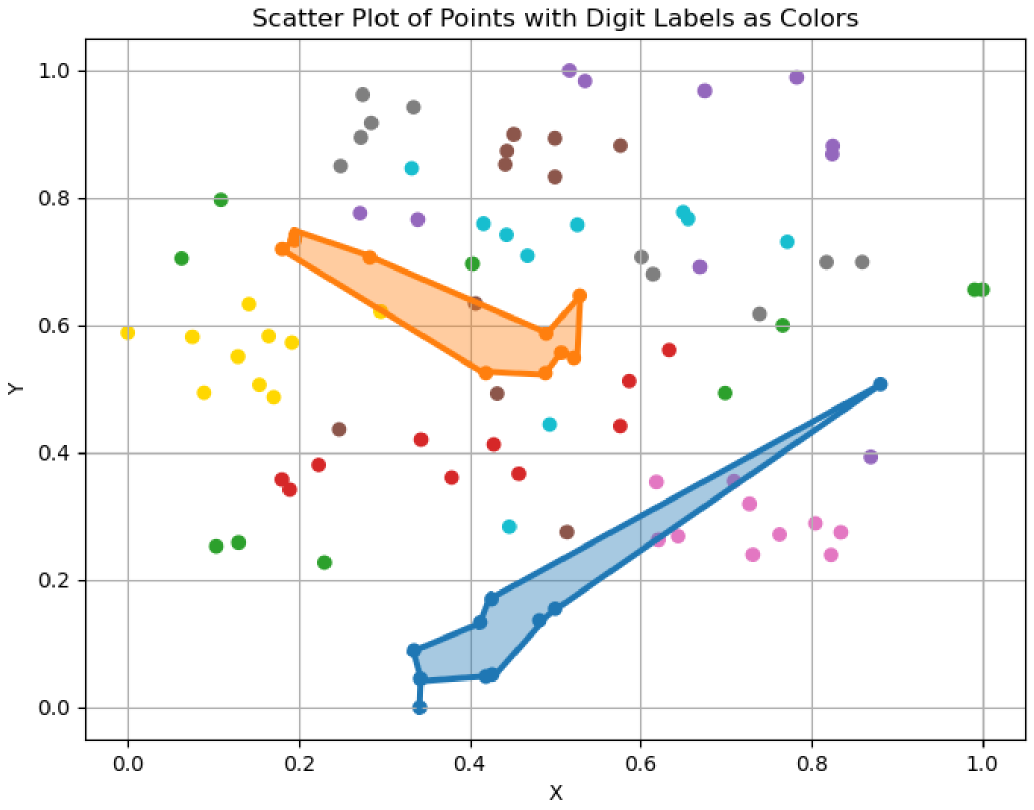
MANUAL MAPPING

For 0



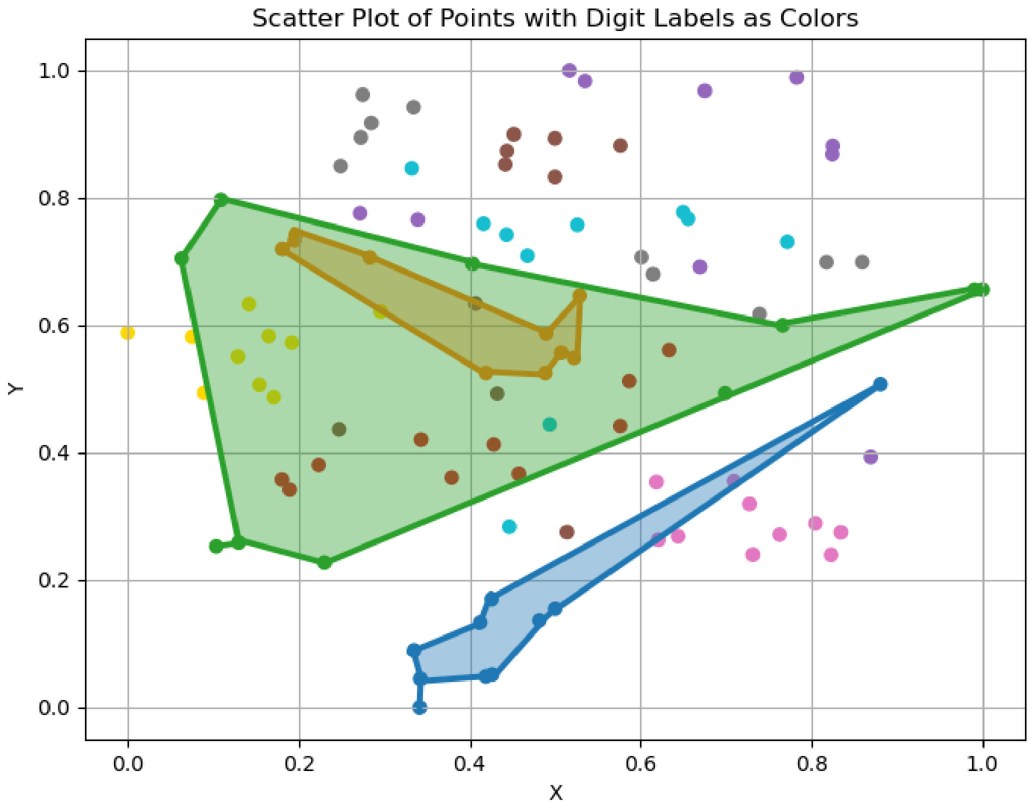
MANUAL MAPPING

For 1



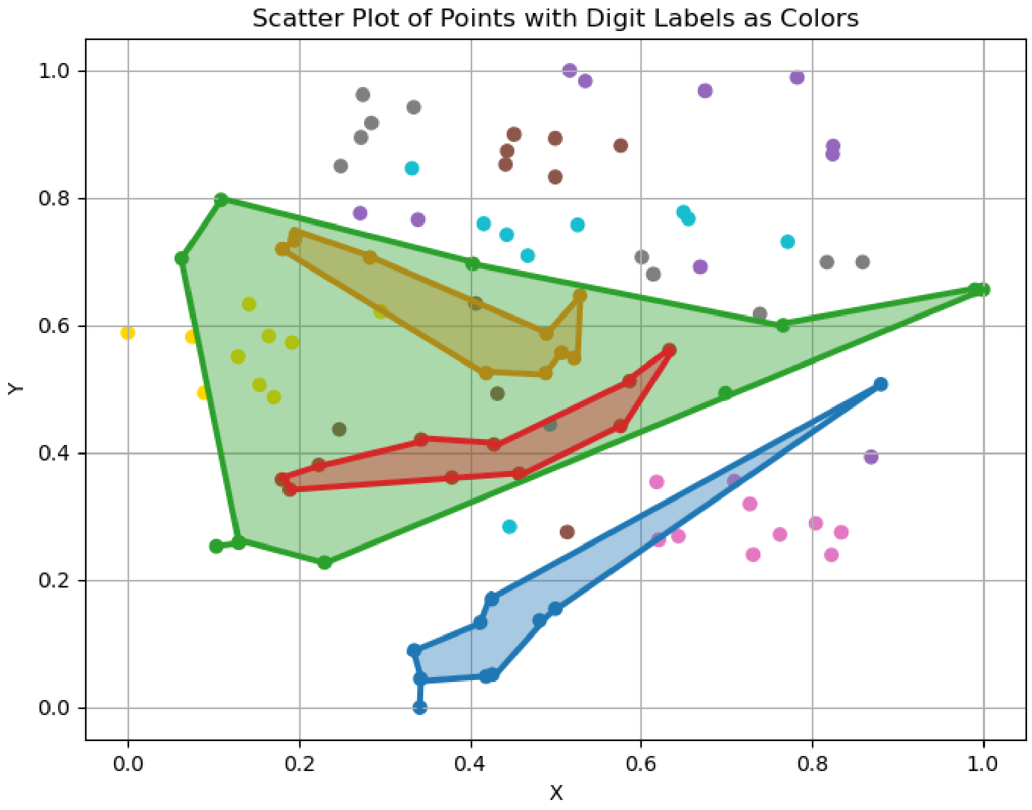
MANUAL MAPPING

For 2



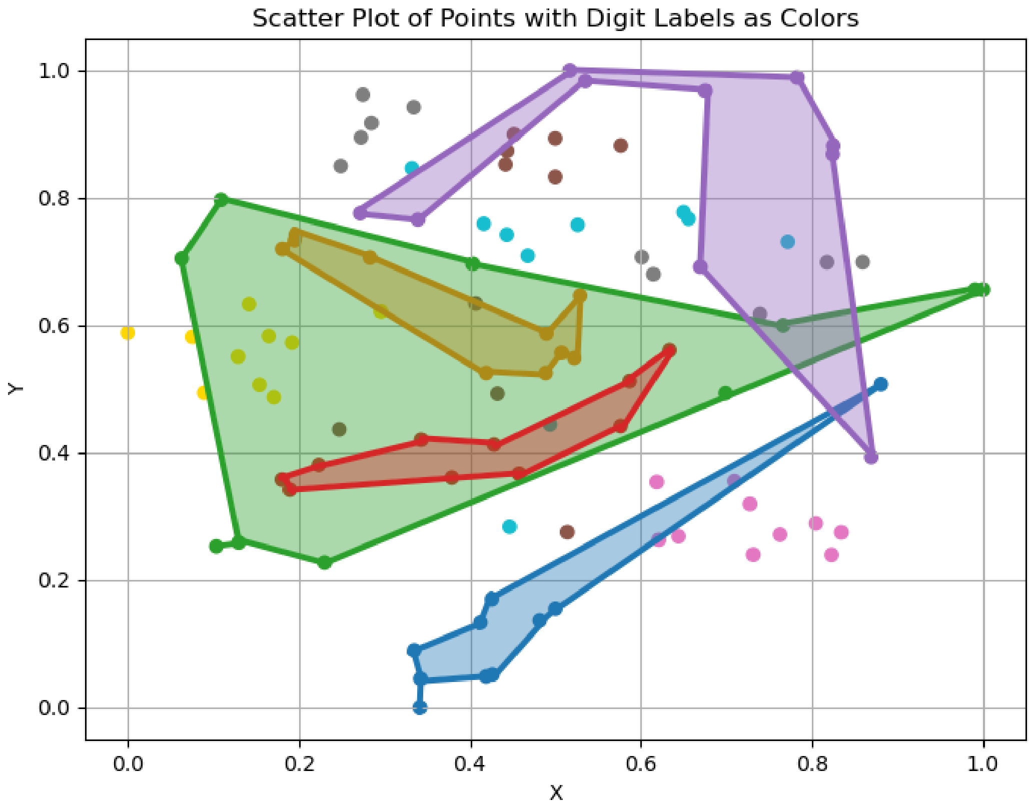
MANUAL MAPPING

For 3



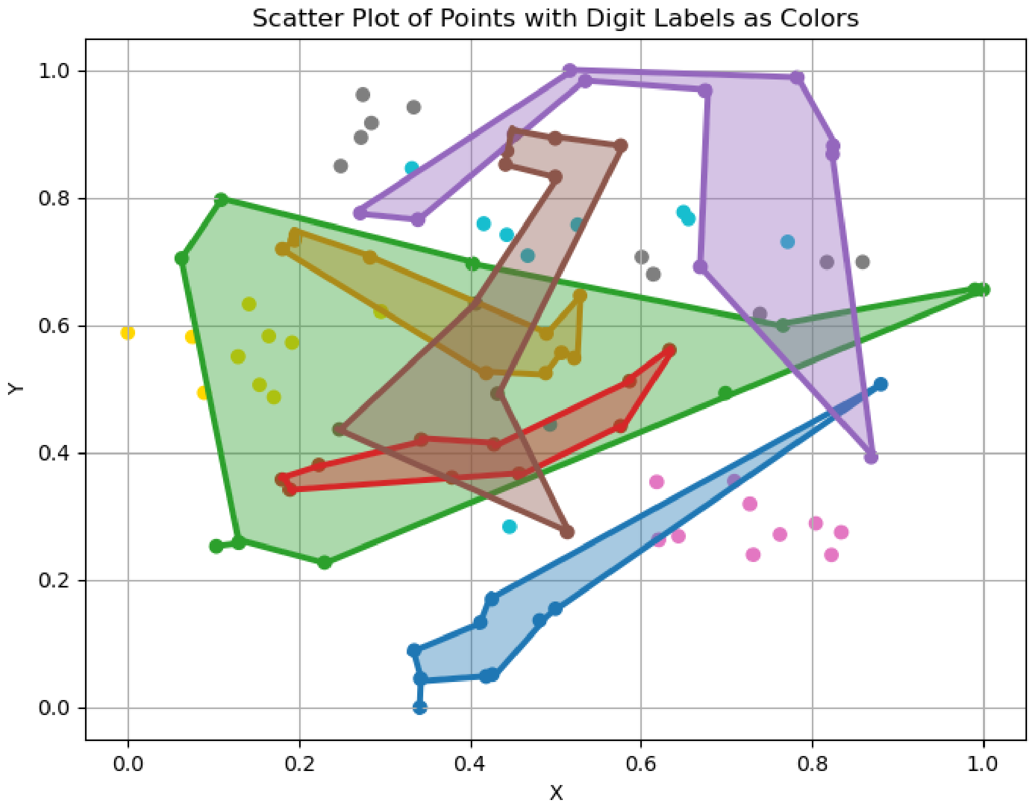
MANUAL MAPPING

For 4



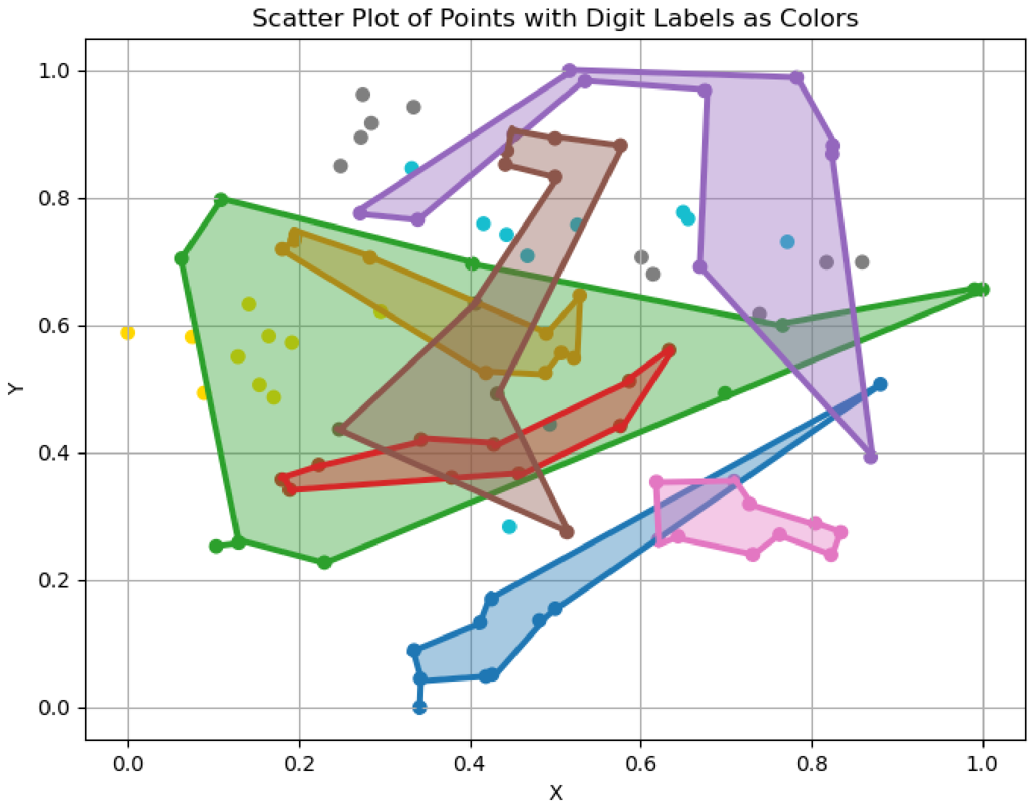
MANUAL MAPPING

For 5



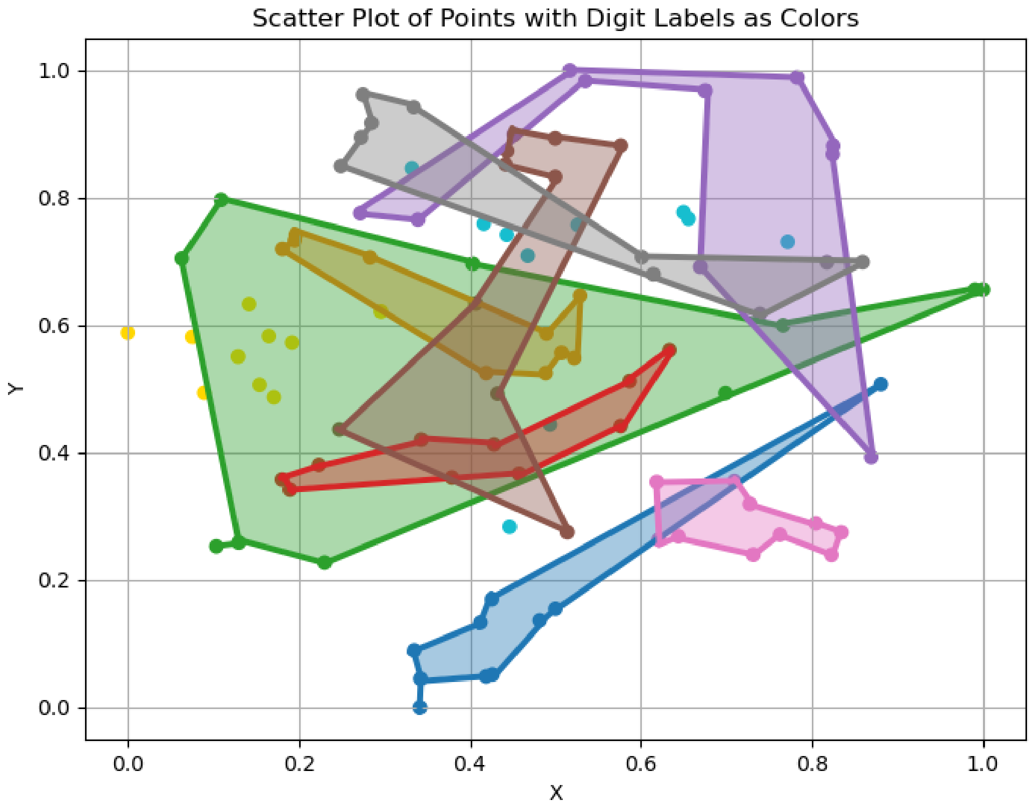
MANUAL MAPPING

For 6



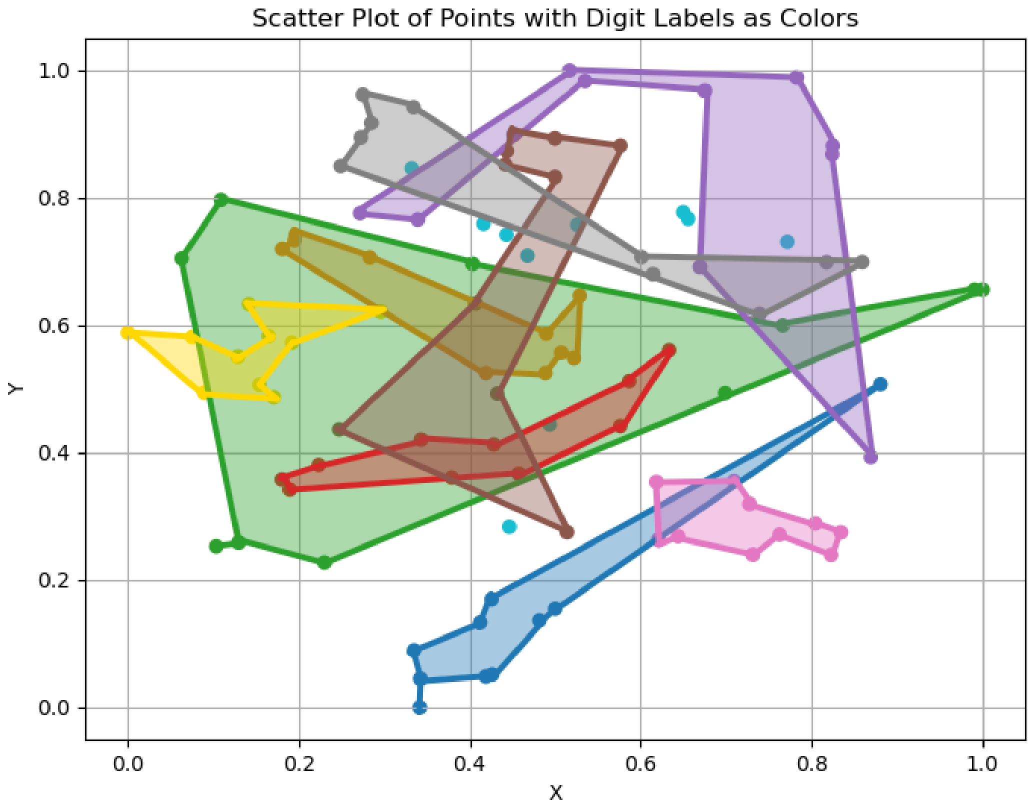
MANUAL MAPPING

For 7



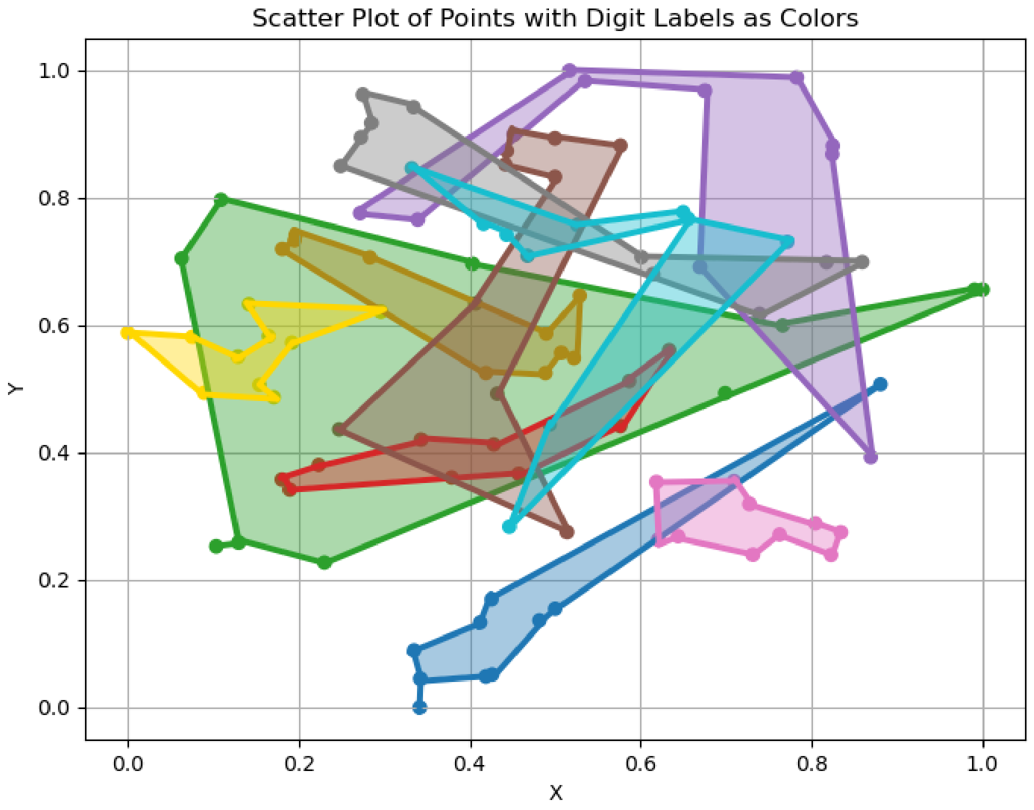
MANUAL MAPPING

For 8



MANUAL MAPPING

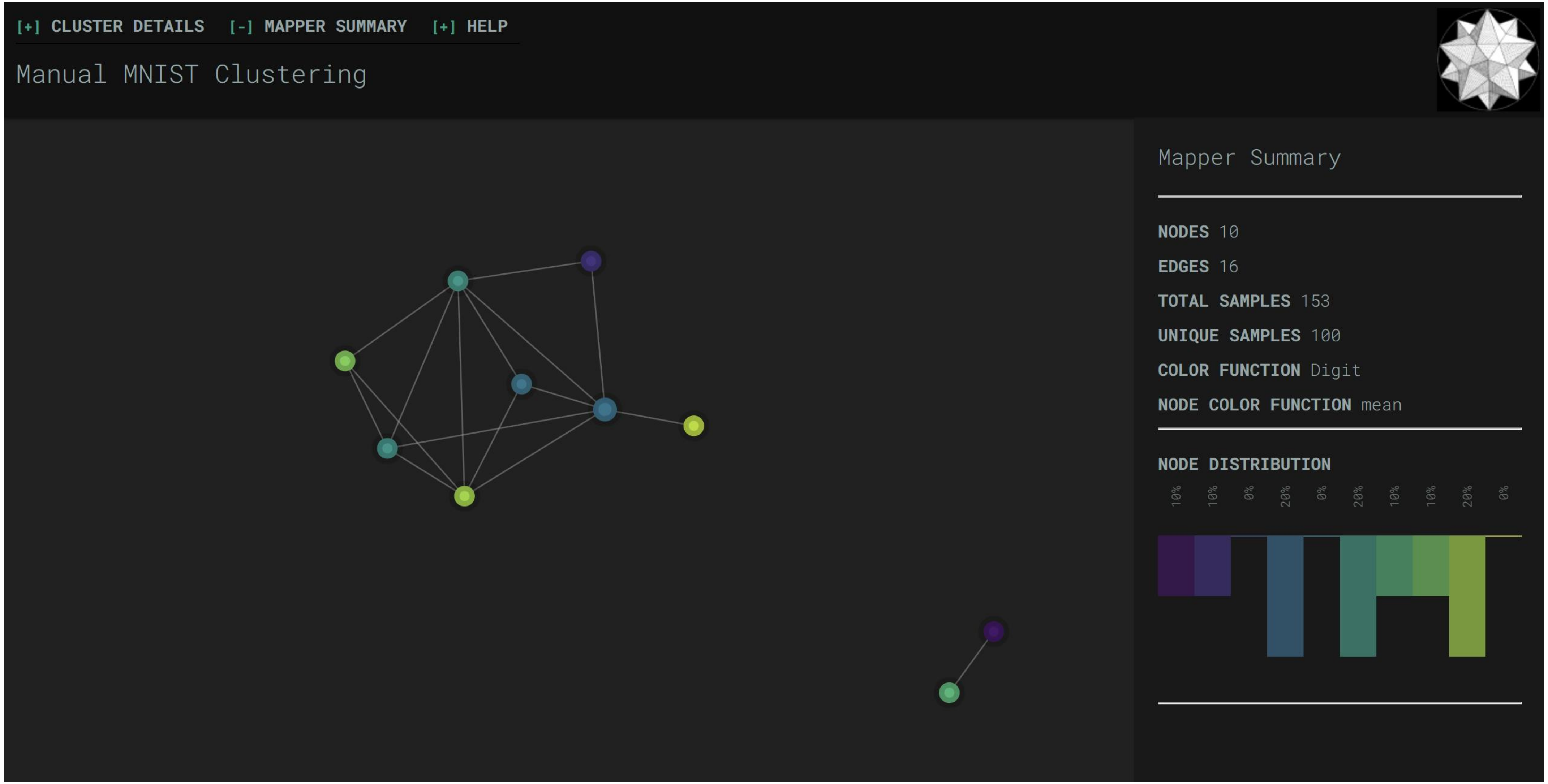
For 9



MANUAL MAPPING

- Cluster for Digit x – concave hull created from all the data points of digit x
- Node for Digit x – contains all data points within the cluster for digit x, even if some data points aren't of digit x
- Edge – formed between two nodes when a data point is in both nodes
- May be improved by calculating edges by cluster overlap instead of data point overlap.

MANUAL MAPPING

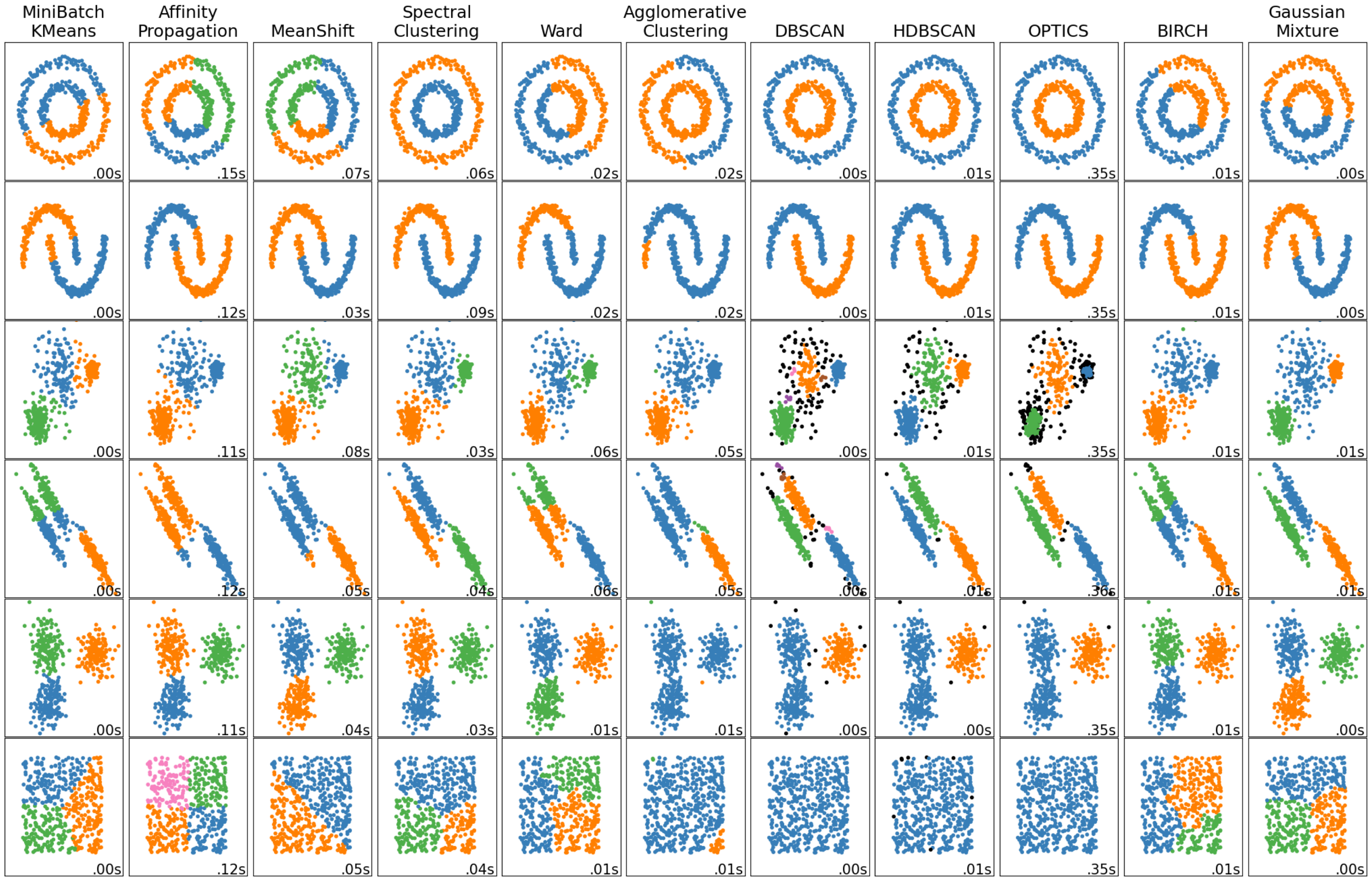


AUTOMATIC MAPPING

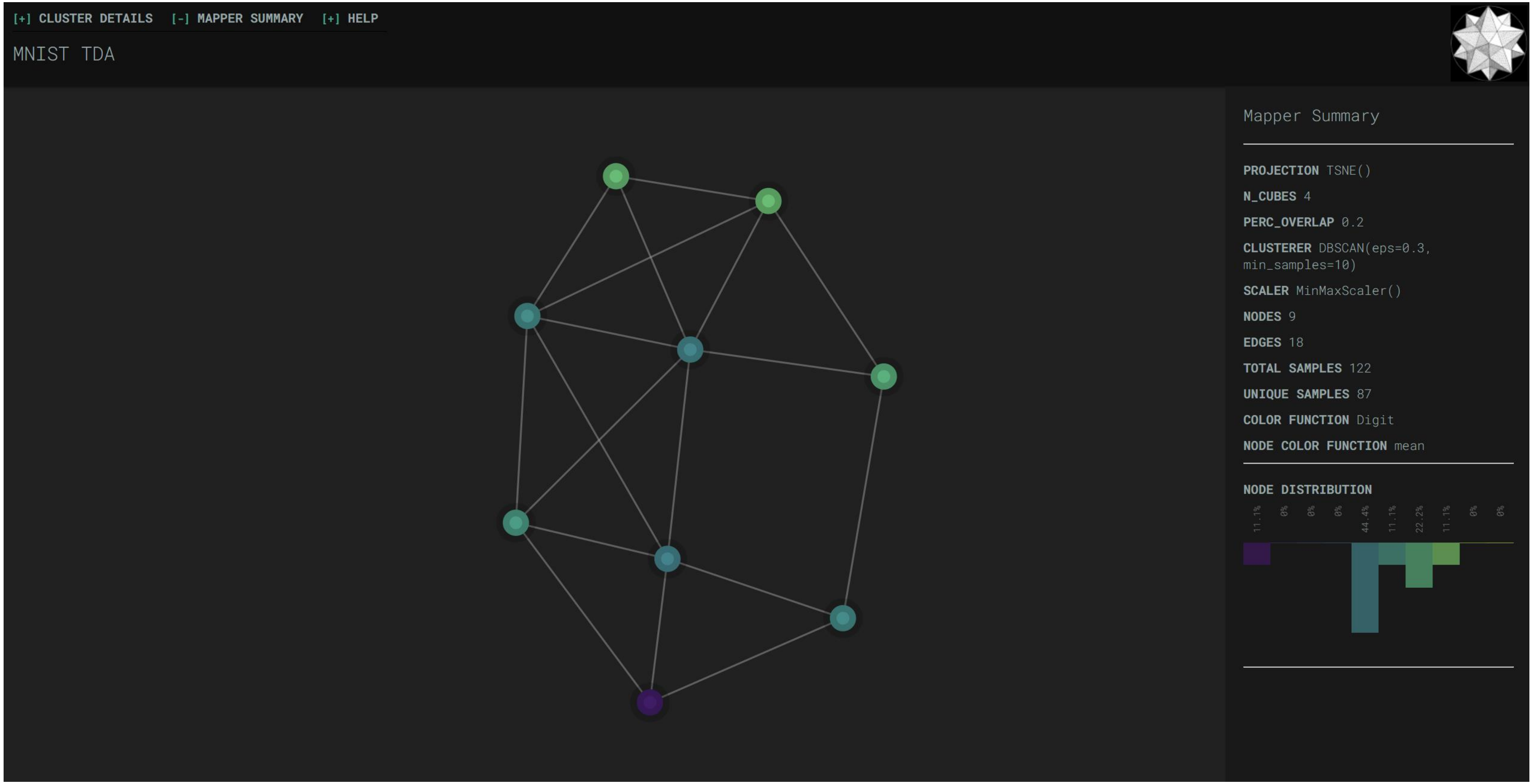
- KeplerMapper provides covering scheme
  - Each dimension spanned by **hypercubes**
  - Adjacent cubes have **% of overlap**
  - Hypercubes per dimension and % of overlap are parameters

AUTOMATIC MAPPING

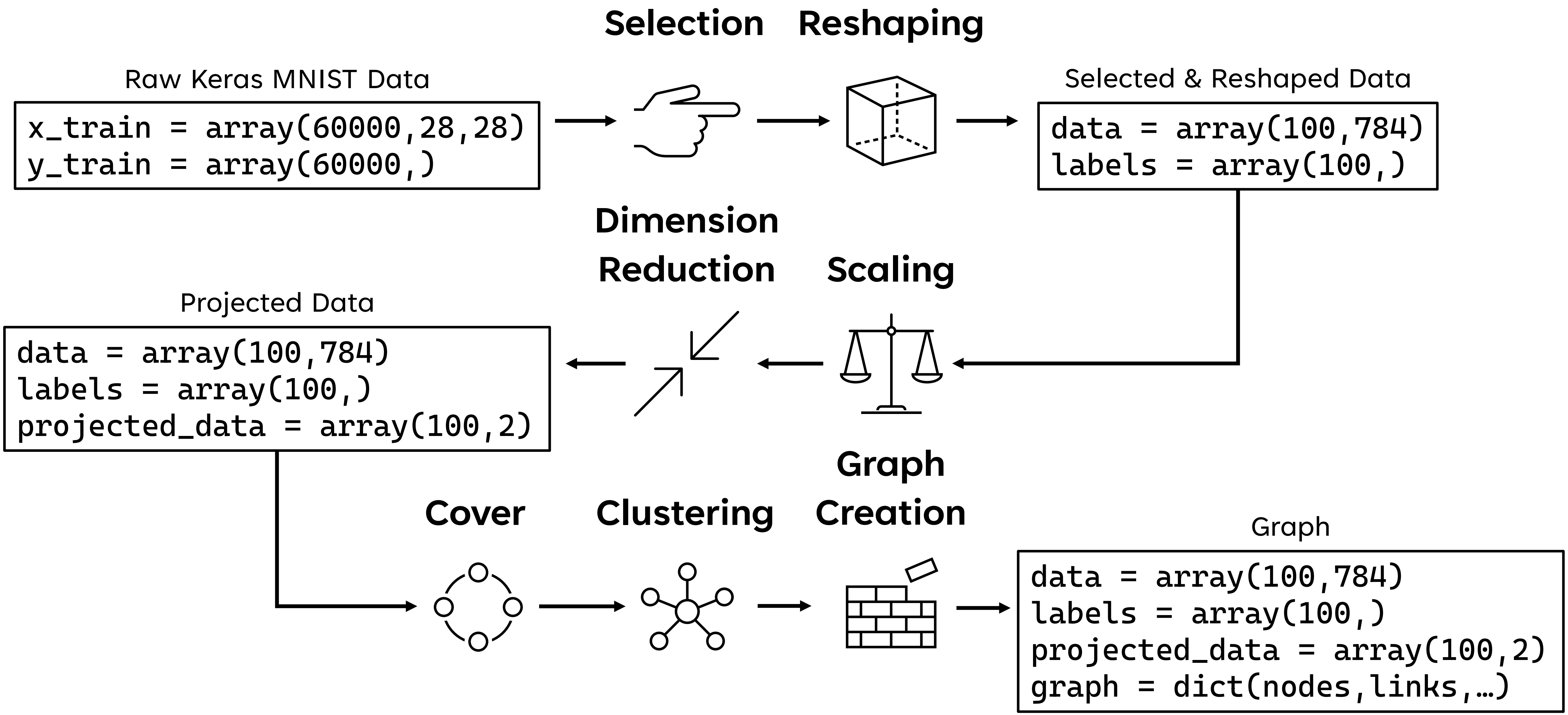
- Many different clustering algorithms, each with different strengths and use cases.
- **DBSCAN** worked best for us.
- **Spectral Clustering** looks promising.



AUTOMATIC MAPPING



DATA PIPELINE



INTERESTING FINDINGS

- 0 and 6 typically close on the scatter plot but no overlap
- 9 has large cluster (similar topology to many digits) and almost never overlap with 0 or 6
- 5 and 4 typically close to 9 on the scatter plot
- Both 1 and 3 typically within 2’s cluster
- 9 and 7 typically close on the scatter plot
- **All of this could change with different/more data, but consistent across lenses/projection pipelines!**

THANK YOU

Interactive HTML files,
Jupyter Notebook w/ code,
Project proposal,
Presentation file, etc.