DBSCAN

Evan Burdett, Isaac Summerford, Carter Williams

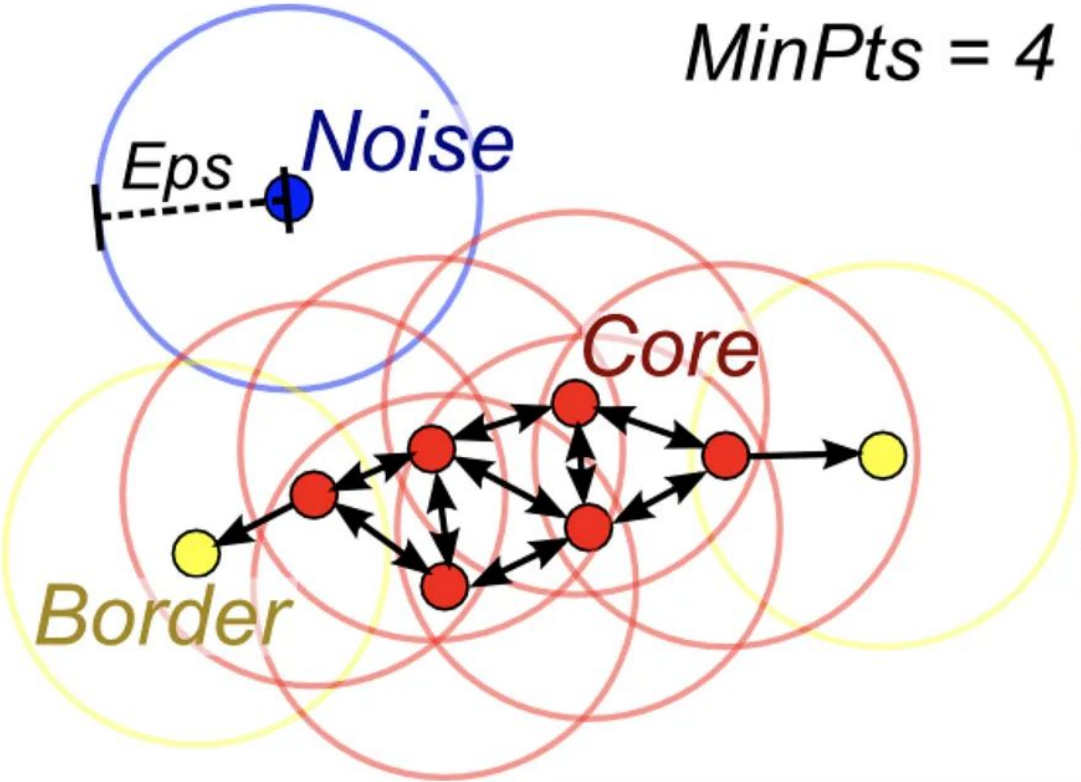
DBSCAN Introduction

- DBSCAN: Density-Based Spatial Clustering of Applications with Noise
- Data clustering algorithm proposed by Ester, Kriegel, Sander, and Xu in their paper published in 1996
- Density Based and Non-Parametric (no specific distribution)
- Given a set of points in some space, closely packed points are grouped together and points in low-density areas are marked as outliers

DBSCAN Introduction: Parameters and Points

- Parameters
  - eps (epsilon): radius of region considered to be a neighborhood
    - If eps is too small, you could end up having too many points be label as noise
    - If eps is too big, separate clusters may merge together and defeat the purpose of the algorithm
  - minPts: minimum number of points within the eps radius to make a neighborhood dense
- Points
  - Core Points: points that are within a dense neighborhood
  - Border Points: points that are close to core points but do not have enough neighbors to be considered core points
  - Outlier Points: points that do not belong to any cluster and are considered noise

DBSCAN Introduction: Parameters and Points



Red: Core Points

Yellow: Border points. Still part of the cluster because it's within epsilon of a core point, but not does not meet the min_points criteria

Blue: Noise point. Not assigned to a cluster

Graphic from: https://elutins.medium.com/dbscan-what-is-it-when-to-use-it-how-to-use-it-8bd506293818

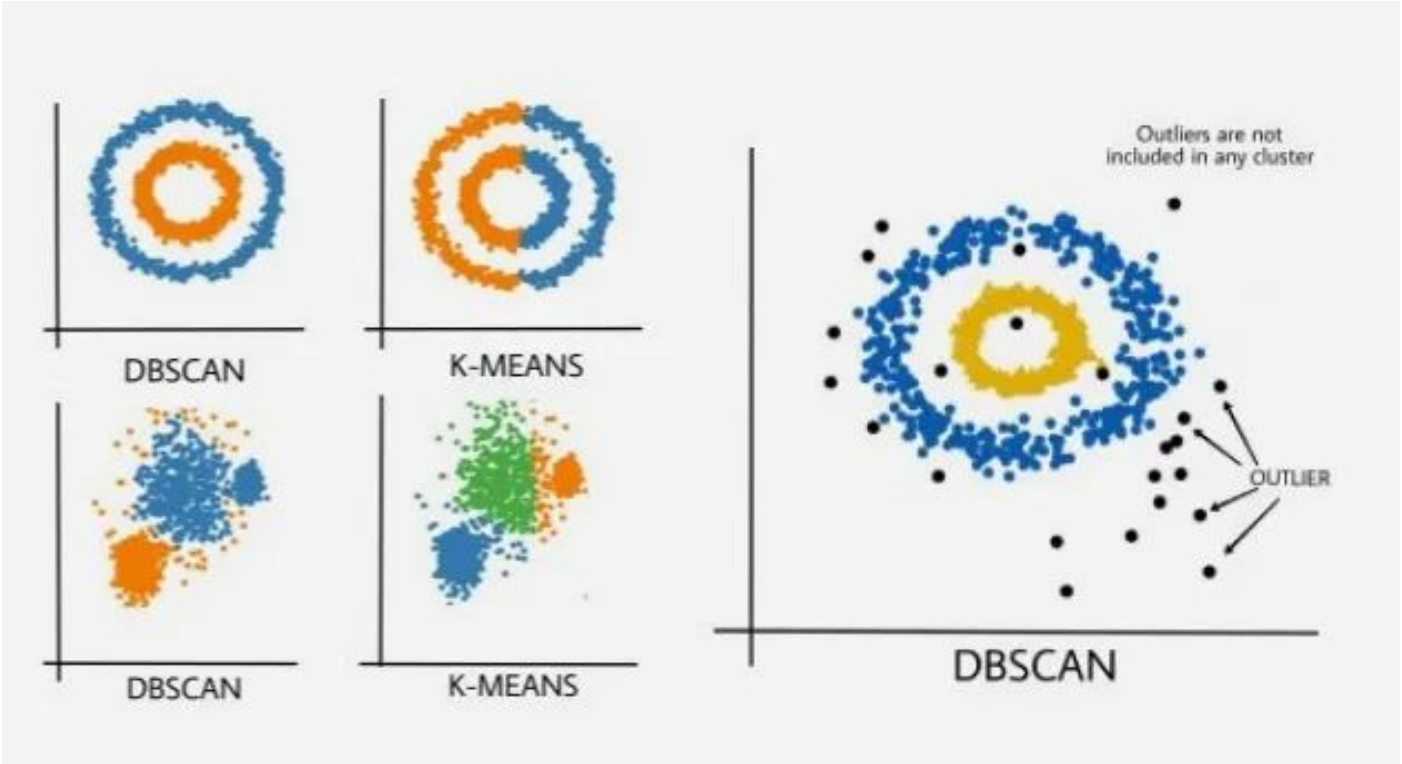
DBSCAN Introduction: Steps

1. Identify Core Points
    a. Iterate through all of the points, counting the number of points within its eps neighborhood, if it is greater than or equal to minPts it is a core point
2. Form Clusters
    a. Make a cluster for every core point that does not have one yet, add all points within the radius of the core point to the cluster
3. Establish Density Connectivity
    a. Two points have density connectivity if there is a chain of points that exists where each point is within the radius of the next point
4. Label Noise Points
    a. All points that are not in a cluster are labeled as noise

DBSCAN Introduction: Advantages and Disadvantages

- Advantages:
  - Do not need to specify the number of clusters like you have to in K-Means
  - It’s good at identifying arbitrarily shaped clusters, other algorithms are only good at identifying spherical clusters
  - Handles noise well when compared to other algorithms
- Disadvantages:
  - Not as good for straightforward and spherical data sets
  - Has two parameters instead of just one in other algorithms

DBSCAN Introduction: DBSCAN vs K-Means



Graphic from: https://www.geeksforgeeks.org/machine-learning/dbscan-clustering-in-ml-density-based-clustering/

Real-World Examples

- Geospatial Analysis
  - Mapping crime incidents in a city to find high-crime “hotspots.”
- Fraud & Anomaly Detection
  - Flagging unusual credit card transactions.
- Customer Segmentation
  - Grouping shoppers by shopping frequency and spending level
- Image Processing
  - Segmenting brain tumors in an MRI scan.
- Medical & Biological Research
  - Clustering genes with similar expression levels across many samples.
- Astronomy
  - Identifying galaxy clusters in deep-space sky surveys.

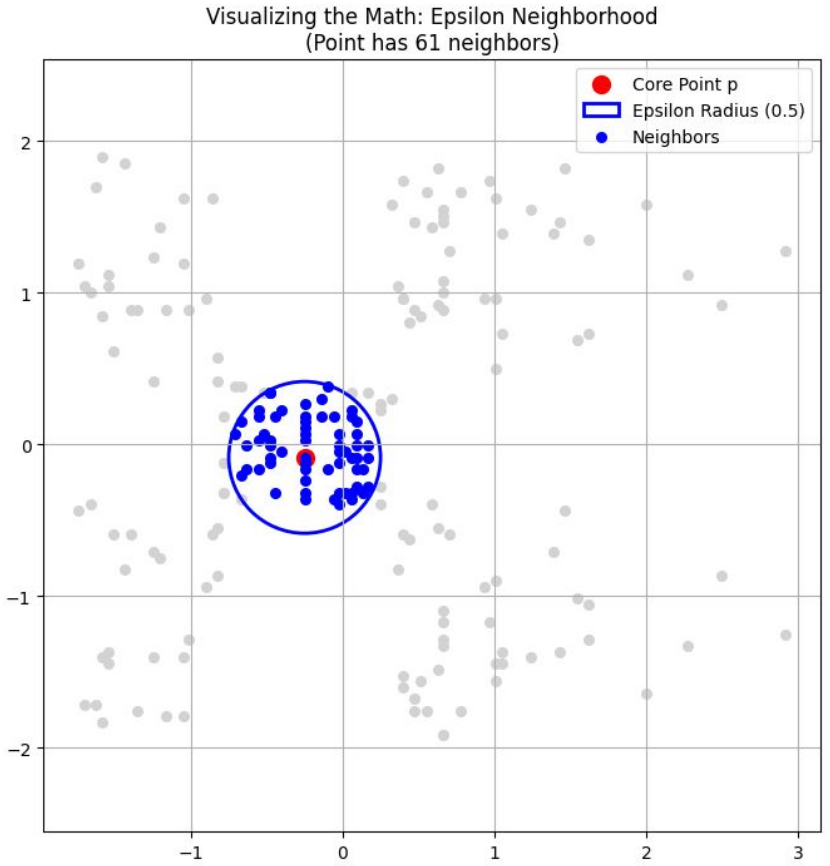
Dataset & Model Design

Dataset: Real-world "Mall Customer" data comparing Annual Income to Spending Score (on a 1-100 scale).

Model: DBSCAN (Density-Based Spatial Clustering).

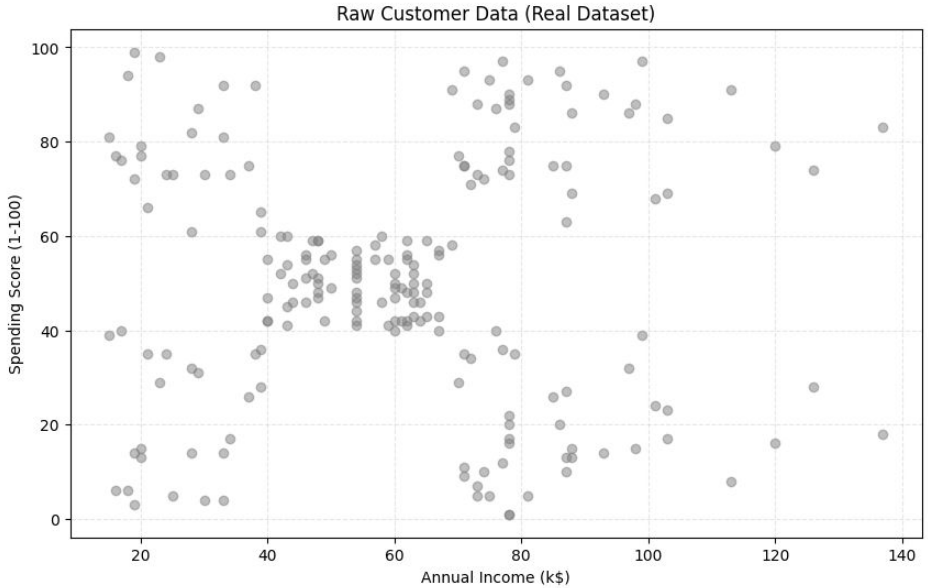
Why this fits our data:

- It automatically labels irregular behavior as "Noise" (outliers).
- It's great for finding natural, non-spherical cluster shapes.
- It discovers the right number of clusters on its own (no need to specify a fixed K).
  Core Logic: Points are grouped by density. A cluster starts when a "Core Point" has at least 5 neighbors (MinPts) within a specific distance (Epsilon).



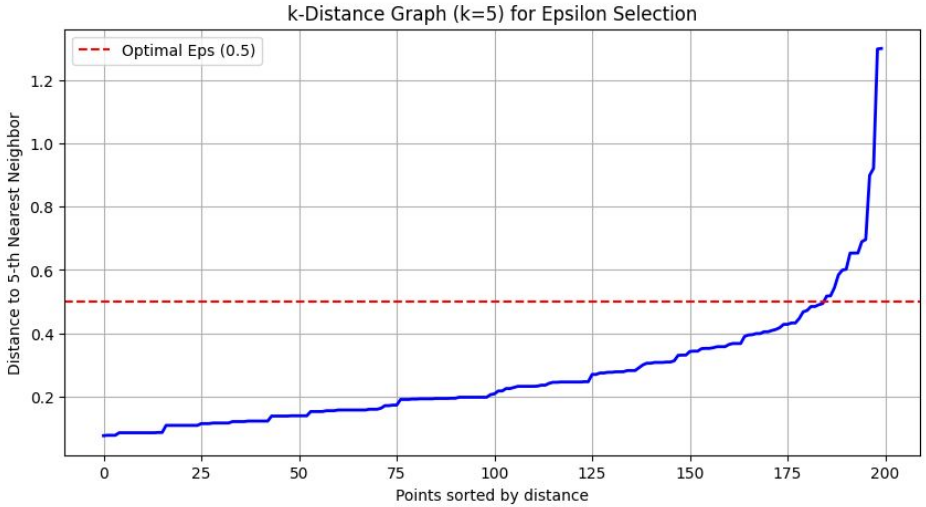
Dataset Overview and Preprocessing

- Dataset: Real-world "Mall Customer" data comparing Annual Income to Spending Score (1–100).
- Preprocessing: Applied StandardScaler to normalize inputs.
  - Income ($15k–$137k) and Score (1–100) have vastly different magnitudes.
  - Without scaling, the Euclidean distance calculation would be entirely biased toward Income.
- Computational Efficiency: Used Scikit-Learn’s implementation utilizing KD-Trees for spatial indexing.
  - Reduces neighbor search complexity from O(N2) to O(NlogN).



Parameter Optimization (Finding Epsilon)

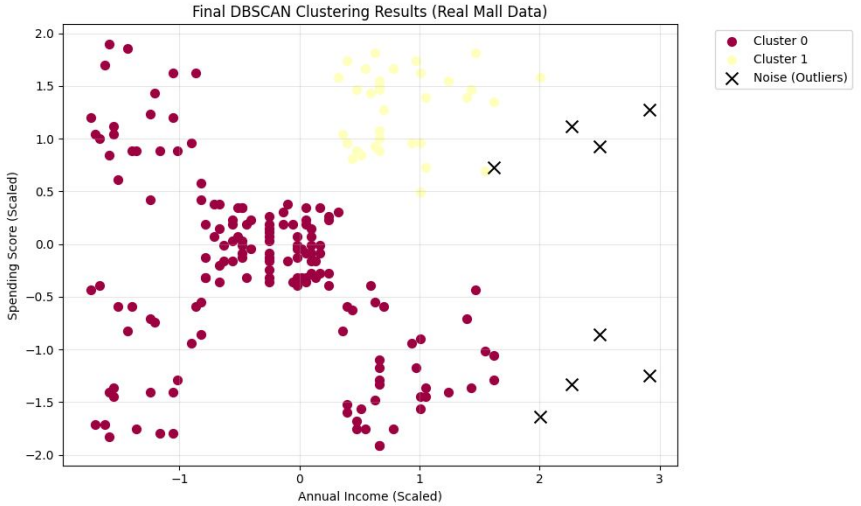
- The Elbow Method: Calculated the distance to the 5th nearest neighbor for every datapoint.
  - Sorted and plotted these distances to visualize the density drop-off
  - The “Elbow” (sharp curve) suggested an initial Epsilon of ~ 0.5.
- Iterative Tuning: We manually adjusted epsilon (0.50 -> 0.40 -> 0.365).
  - The goal was to resolve the “Bridge Effect,” where distinct clusters were merged due to density gradients.
  - Result: An epsilon of 0.365 provided the cleanest separation between clusters.

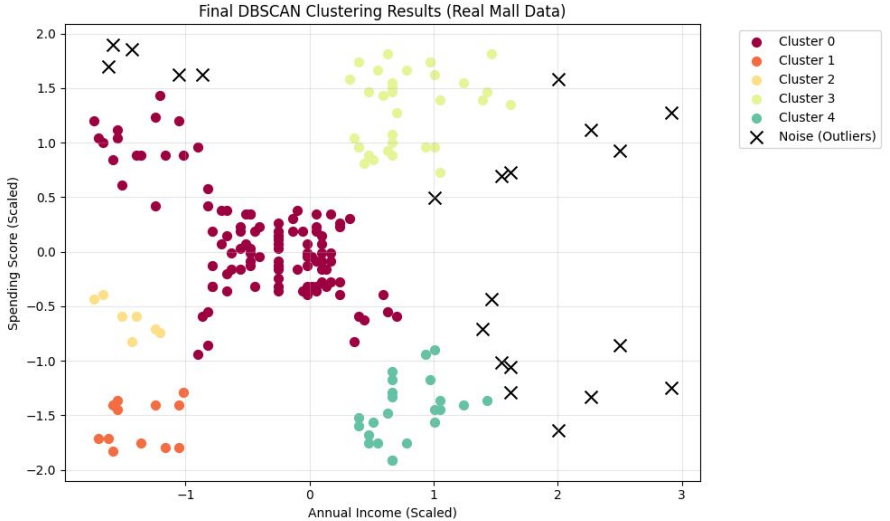


Clustering Results & Evaluation

We evaluated the results using a few key criteria:

- Qualitative Inspection: Since we lacked true labels, we focused on how easily we could interpret the resulting clusters. For example, did shoppers with "High Income/Low Spend" form a clear, distinct group? 5 natural segments.
- Noise Ratio: We monitored the percentage of points labeled as X (Noise).
  - A ratio that was too high (>20%) suggested Epsilon was too strict.
  - A ratio that was too low (<1%) suggested Epsilon was too loose, potentially merging distinct clusters.
  - Our final ratio was 10.50% (21 outliers)
- Cluster Count: We verified if the model successfully identified the 5 natural segments we had observed in the data's distribution.

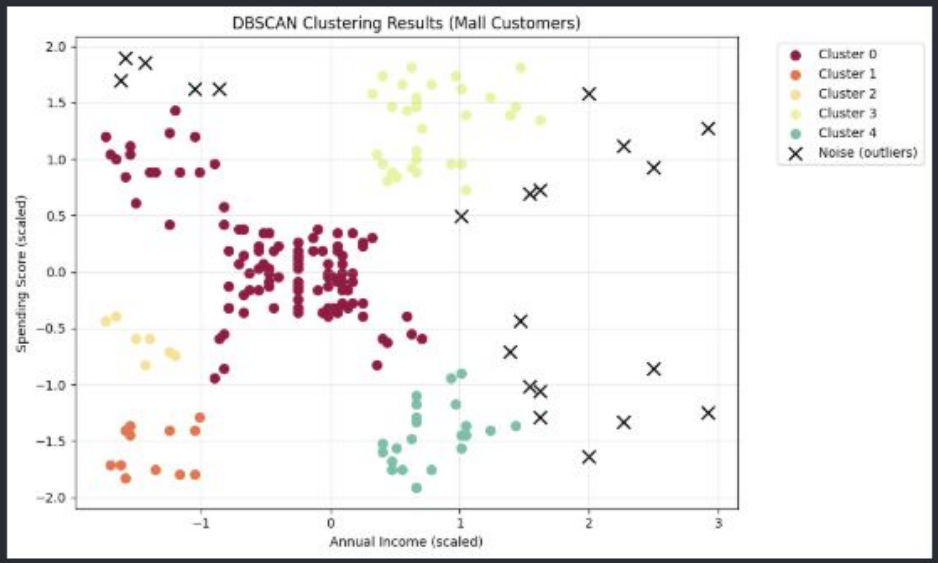




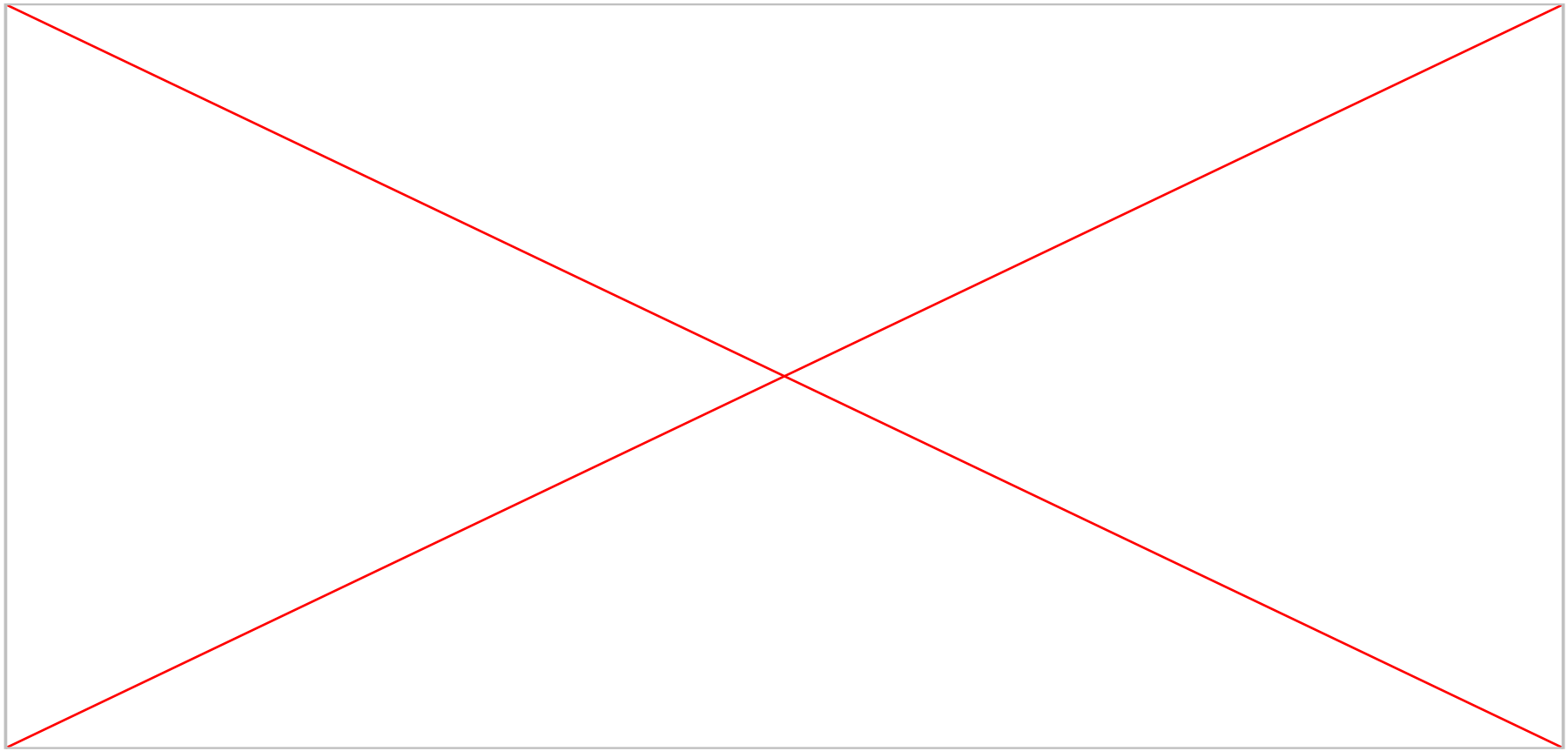
Our Code

```python
# Final DBSCAN run and evaluation

db_final = DBSCAN( (variable) labels_: ndarray[_AnyShape, dtype[Any]]
labels = db_final.labels_

n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
n_noise = list(labels).count(-1)
noise_ratio = n_noise / len(labels)

print("Results:")
print(f"Clusters found: {n_clusters}")
print(f"Noise points (outliers): {n_noise}")
print(f"Noise ratio: {noise_ratio:.2%}")

plt.figure(figsize=(10, 6))

unique_labels = set(labels)
colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1, len(unique_labels))]

for k_label, col in zip(unique_labels, colors):
    if k_label == -1:
        col = [0, 0, 0, 1]
        label_text = "Noise (outliers)"
        marker_shape = "x"
        size = 100
    else:
        label_text = f"Cluster {k_label}"
        marker_shape = "o"
        size = 50

        # nothing special, same col

    class_member_mask = labels == k_label
    xy = X_scaled[class_member_mask]
    plt.scatter(
        xy[:, 0], xy[:, 1], c=[col], marker=marker_shape, s=size, label=label_text
    )

plt.title("DBSCAN Clustering Results (Mall Customers)")
plt.xlabel("Annual Income (scaled)")
plt.ylabel("Spending Score (scaled)")
plt.legend(bbox_to_anchor=(1.05, 1), loc="upper left")
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

```
Results:
Clusters found: 5
Noise points (outliers): 21
Noise ratio: 10.50%
```



Demo of Our Code



Sources

GeeksforGeeks. “DBSCAN Clustering in ML Density Based Clustering.” *GeeksforGeeks*, 6 May 2019, www.geeksforgeeks.org/machine-learning/dbscan-clustering-in-ml-density-based-clustering/.

Lutins, Evan. “DBSCAN: What Is It? When to Use It? How to Use It.” *Medium*, 4 Dec. 2020, elutins.medium.com/dbscan-what-is-it-when-to-use-it-how-to-use-it-8bd506293818.

Wikipedia Contributors. “DBSCAN.” *Wikipedia*, Wikimedia Foundation, 16 July 2019, en.wikipedia.org/wiki/DBSCAN.