BA820 – Project M4

Cover Page

- **Project Title:** Exploring Latent Color Patterns and Similarity in Bob Ross Paintings
- **Section and Team Number:** B1 Team 3
- **Team Members:** Yu-Jiun Zou

1. Refined Problem Statement & Focus (~0.5 page)

In this milestone I continue to focus on the same question: “What are the potential structures in Bob Ross paintings based on how paint colors are combined, and do these structures reveal distinct palette styles?”. The core domain question remains unchanged, during M2 and M3 our analysis focused on color combinations with clustering and association analyses, it shows that most paintings share a highly similar core palette, the results revealed one dominant cluster containing the majority of paintings, with smaller clusters reflecting differences in emphasis rather than fundamentally different palette types, that the assumptions of there are independent stylistic was not supported.

These findings shifted the focus. Instead of continuing to search for clearly distinct palette styles, I kept experimenting with different clustering methods to see whether it is possible to avoid a structure dominated by one overwhelmingly large cluster. At the same time, the refinement now centers on understanding how variation occurs within this largely consistent color system. To further explore this, I included title-based text analysis to examine thematic patterns within clusters. Therefore, while the central question remains the same, the focus has shifted from trying to identify clearly separated stylistic categories to better understanding how title themes and color usage relate to each other.

2. EDA & Preprocessing: Updates (~0.5 page)

Earlier EDA showed that Bob Ross paintings share a highly consistent core palette, as the Jaccard similarity between paintings was generally high, and clustering results revealed one dominant cluster containing most paintings. Specifically the cluster-level heatmap (Figure 1) further showed the differences in color emphasis across clusters, it raised the question of whether these variations might align with thematic elements in the painting titles. This motivated a deeper exploration of what distinguishes clusters beyond color usage alone. Since the previous clustering showed limited separation in color structure, I applied DBSCAN with different parameters to re-group the paintings and then integrated title-based text analysis to examine whether stronger themes and color patterns could be identified.

No changes were made to the original preprocessing pipeline and EDA steps, as they included binary encoding and similarity computation that provided key justification for clustering and association analysis. However in M4, an additional preprocessing step was introduced for painting titles (Figure 2). Titles were cleaned and transformed into structured word representations to enable comparison of thematic patterns across clusters. Extremely rare words were filtered using a minimum document frequency threshold to improve statistical stability and reduce noise in cluster-level comparisons (Figure 3). These additional preprocessing steps were necessary to enable semantic comparison across clusters while maintaining alignment with earlier color-based analysis.

3. Analysis & Experiments (~2 pages)

Method1: DBSCAN

DBSCAN was introduced to test whether a density-based approach can reveal more nuanced structure, using Jaccard distance is appropriate because it emphasizes overlap in shared used paints. DBSCAN is suitable because it can model a dense core region plus smaller dense and noise points, helps to answer whether meaningful palette patterns exist beyond a single mega-cluster and whether those patterns correspond to distinct paint-combination tendencies.

Before running DBSCAN, I examined global paint usage rates and found that a small set of paints appeared in almost every painting (e.g., *Titanium White*, *Alizarin Crimson*, *Van Dyke Brown*, each >90%) (Figure 4). To evaluate this, I compared the pairwise Jaccard distance distributions before and after removing the top three most frequent paints (Figure 5). After removal, the distance distribution shifted right and became more dispersed, and the mean pairwise distance increased (from ~0.365 to ~0.448). This indicates improved differentiation among paintings and supports the decision to remove those base colors so DBSCAN can focus on more discriminative combinations rather than universally shared paints.

I used a k-distance plot (Figure 6) to guide the selection of eps, testing min_samples values of 3, 5, and 8. The plot suggests a reasonable eps range of approximately 0.10 to 0.15. Instead of showing a sharp elbow, the curve increases gradually with a steeper rise in the tail, indicating that the palette space does not form clearly separated dense regions but rather follows a more continuous structure. To choose a configuration that is both interpretable and stable, I filtered n_clusters between 3 and 8, not to force many tiny clusters but to obtain a small number of interpretable palette variants beyond the core. noise_rate < 0.30, as DBSCAN can label boundary points as noise, setting a threshold keeps the result informative while still allowing boundary cases. silhouette > 0.40, because earlier evidence suggests only moderate separation, a silhouette threshold ensures clusters have at least meaningful separation among retained points. Very high silhouette values in some runs can occur for trivial reasons, such as many tiny clusters or excessive noise, so silhouette was used as a screening metric not the main objective.

Final choose with eps = 0.15 and min_samples = 5, DBSCAN identified one major core palette (Cluster 2, n = 257) and three smaller variants (Cluster 1: n = 48; Cluster 0: n = 9; Cluster 3: n = 8), while 81 paintings were labeled as noise (Figure 7). The large cluster represents the shared baseline palette across most paintings. The smaller clusters reflect emphasis-based variations rather than entirely distinct palette systems. After clustering, I used lift to identify which colors appear disproportionately often in each cluster compared to the global baseline, helping interpret the stylistic emphasis of each cluster (Figure 10).

One surprising aspect is that although Cluster 2 remains relatively large, the overall distribution is much more balanced than in the earlier hierarchical clustering results, where one cluster dominated nearly the entire dataset. DBSCAN provides a clearer separation between the core palette and smaller variants, and paint-level lift further shows more representative and differentiated color patterns across groups, making the interpretations more meaningful. Finally, I examined the 84 noise paintings, which represent a meaningful portion of the dataset. Among them, 38 are closest to Cluster 3 and 22 to Cluster 2, with a median distance of about 0.47 (Figure 8), suggesting they reflect boundary or low-density variations near existing clusters rather than completely separate palette styles.

Method2: Bag-of-Words (BoW)representation of titles with Cosine Similarity

To examine whether thematic patterns align with palette-based clusters, I applied a Bag-of-Words (BoW) representation to painting titles. This method is appropriate because Bob Ross titles are typically short and concise and since word order carries limited additional meaning in these short titles, word presence is more informative than sequence, making BoW a suitable representation.

Before directly applying it, I ran the diagnosis and applied a min_df threshold to remove noise. Since with min_df = 1, more than half (57%) of the words appeared only once, which made lift results noisy and unstable (Figure 3). Setting min_df = 2 helped filter out one-time words and produce more reliable comparisons.

To interpret each DBSCAN cluster in the title space, I computed word-level lift. For each cluster, I calculated the fraction of titles containing each word and compared it to the word's global rate across all titles. Lift is more informative than cluster means because it highlights words that are disproportionately associated with a cluster rather than words that are common everywhere. However, lift can become exaggerated when the global base rate is extremely small. To control this, I applied a global frequency filter min_global and tested 0.01 and 0.02. With min_global = 0.01, several low-frequency words produced extreme lift values (e.g., >13), which were hard to interpret and likely driven by small counts. Increasing to min_global = 0.02 removed extremely rare terms and resulted in more stable, interpretable lift patterns. Therefore, I used min_global = 0.02 in the final analysis.

After looking at the top words in each cluster, I computed cosine similarity between cluster-level average BoW vectors to measure how similar clusters are in terms of themes. This produces a similarity matrix shown as a heatmap. The resulting heatmap (Figure 9) shows that clusters 0, 1, and 2 are moderately similar, suggesting they represent variations within a shared theme rather than distinct categories. Cluster 3 appears more distinct, but the separation is not sharp. This is a surprising finding that even after refining DBSCAN thresholds, clusters still show thematic overlap. Certain keywords (e.g., *winter*, *stream*, *valley*) appear more often in certain clusters, but overall the themes are not clearly separated. Instead, they seem to shift gradually from one cluster to another. This suggests that the painting titles follow a continuous pattern rather than forming completely distinct thematic groups.

4. Findings & Interpretations (~1 page)

Across both color-based clustering (DBSCAN) and title-based text analysis (BoW), the results suggest that Bob Ross Bob Ross paintings do not form sharply separated stylistic categories. Instead, they share a strong and consistent core palette, across different clustering approaches, a large central group consistently appeared, showing that most paintings rely on a similar set of foundational colors. This indicates a stable and coherent artistic structure rather than multiple independent styles.

The DBSCAN results reveal more nuance within this structure. Although Cluster 2(n = 257) remains relatively large, the distribution is more balanced compared to the earlier hierarchical clustering results. Cluster 2 represents a shared baseline palette that appears throughout the paintings. Instead of sharply separated artistic categories, most paintings follow a common color system. However, several smaller clusters do emerge, these clusters are not random splits, they show concentrated color tendencies. For example, Cluster 0 emphasizes green tones, Cluster 3 highlights earth tones such as Burnt Umber, and Cluster 1 exhibits stronger blue dominance (see Table 1 for detailed color and keyword distributions).

The noise analysis further supports this interpretation. The paintings labeled as noise are not random outliers or completely unrelated points. Instead, they tend to remain closer to certain clusters and appear to represent low-density or boundary variations within the overall palette structure. This suggests that stylistic differences are gradual rather than sharply separated, and even the more unusual paintings are still structurally connected to the broader visual framework.

When examining the painting titles, the thematic content generally aligns with the color patterns of each cluster. The dominant cluster is associated with general landscape words reinforcing its role as the core style. While the smaller clusters display more focused thematic signals, for example, the darker blue cluster strongly aligns with “winter” and “cabin”. This alignment between color emphasis and title themes suggests that the clusters capture meaningful stylistic tendencies rather than arbitrary groupings. However, the cosine similarity heatmap (Figure 9) shows moderate overlap across clusters, indicating that the themes are not completely separated, they shift gradually across groups. Together, both color usage and title themes suggest variation within a shared underlying structure rather than clearly divided stylistic categories.

To summarize, the results suggest that Bob Ross’s paintings follow a consistent visual system with controlled variation. The differences across paintings are mainly shifts in emphasis rather than completely separate styles. Instead of forming clearly divided categories, the paintings appear to vary gradually within a shared structure. Overall, this shows how strong consistency and subtle variation can exist at the same time within a recognizable and cohesive body of work.

Appendix

Shared GitHub Repository

Link: https://github.com/yujiunzou/BA820-Unsupervised-ML-Project

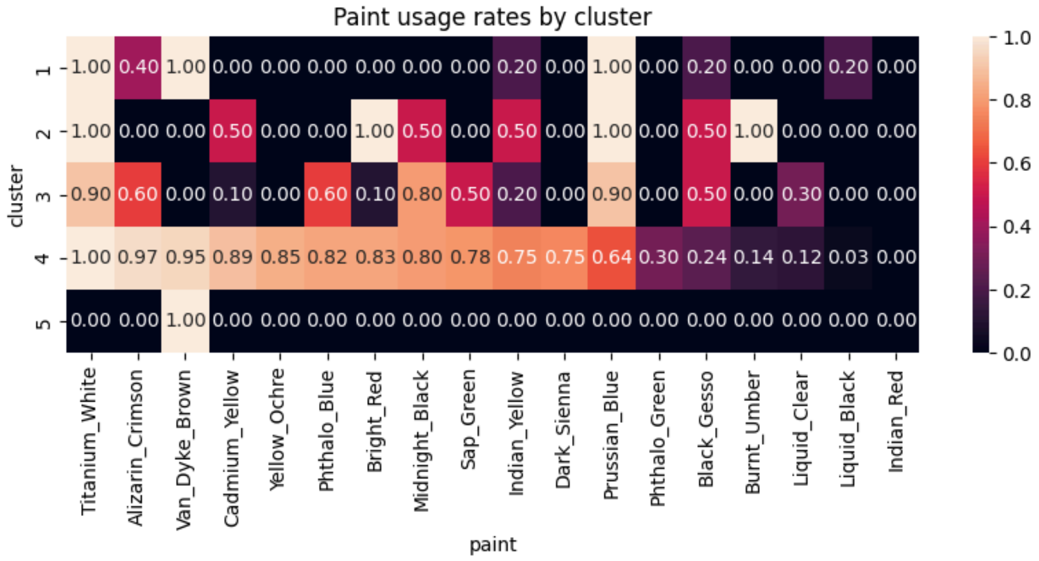
M4 work (where to find the files):

- Project Name: BA820-Unsupervised-ML-Project
- Branch: “*Main*”
- Folder: “*m4_individual_refinement*”
- Folder: “*M4_Yujiun_Zou*”
- Primary notebook: “*M4_Bob_Ross_Paintings_Yujiun_Zou.ipynb*”
- Support file (report PDF): “*M4_Report_Yujiun_Zou.pdf*”

Notebook Link:
https://github.com/yujiunzou/BA820-Unsupervised-ML-Project/blob/main/m4_individual_refinement/M4_Yujiun_Zou/M4_Bob_Ross_Paintings_Yujiun_Zou.ipynb

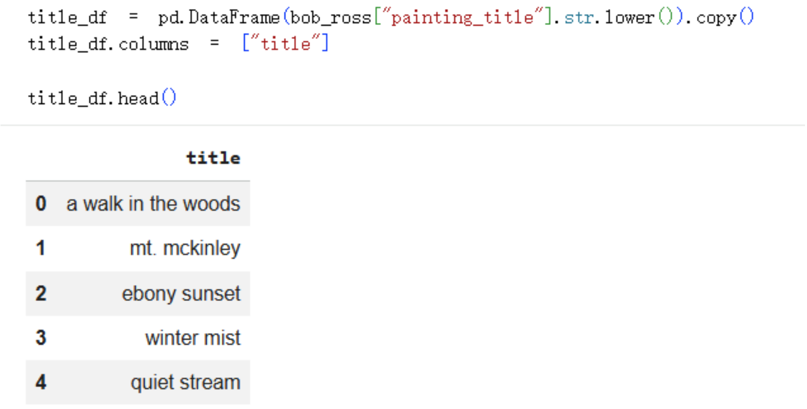
Supplemental Material (Highly Recommended)

Figure 1: Cluster-level paint usage heatmap



cluster	Titanium_White	Alizarin_Crimson	Van_Dyke_Brown	Cadmium_Yellow	Yellow_Ochre	Phthalo_Blue	Bright_Red	Midnight_Black	Sap_Green	Indian_Yellow	Dark_Sienna	Prussian_Blue	Phthalo_Green	Black_Gesso	Burnt_Umber	Liquid_Clear	Liquid_Black	Indian_Red
1	1.00	0.40	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.00	1.00	0.00	0.20	0.00	0.00	0.20	0.00
2	1.00	0.00	0.00	0.50	0.00	0.00	1.00	0.50	0.00	0.50	0.00	1.00	0.00	0.50	1.00	0.00	0.00	0.00
3	0.90	0.60	0.00	0.10	0.00	0.60	0.10	0.80	0.50	0.20	0.00	0.90	0.00	0.50	0.00	0.30	0.00	0.00
4	1.00	0.97	0.95	0.89	0.85	0.82	0.83	0.80	0.78	0.75	0.75	0.64	0.30	0.24	0.14	0.12	0.03	0.00
5	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

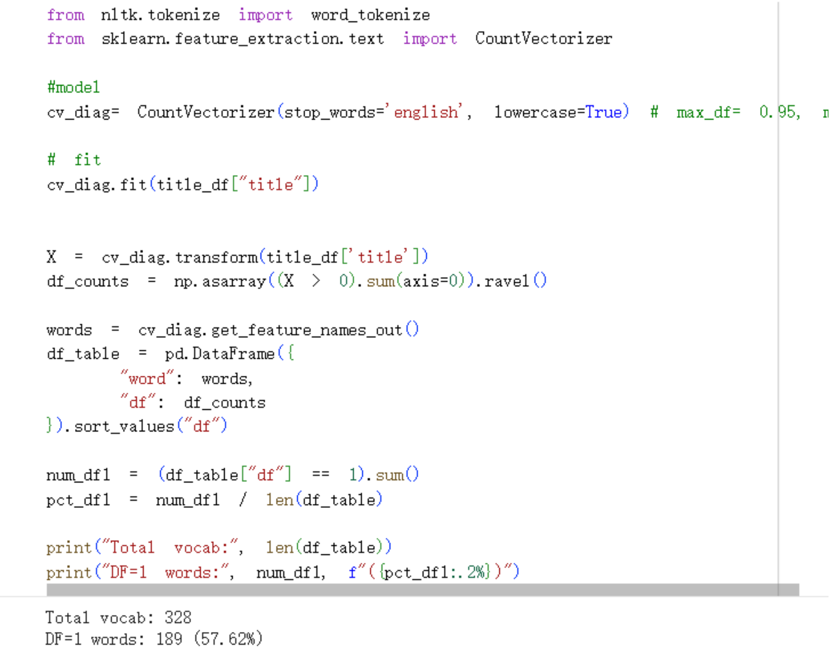
Figure 2:   Title Text Preprocessing



```
title_df  =  pd.DataFrame(bob_ross["painting_title"].str.lower()).copy()
title_df.columns  =  ["title"]

title_df.head()
```

	title
0	a walk in the woods
1	mt. mckinley
2	ebony sunset
3	winter mist
4	quiet stream

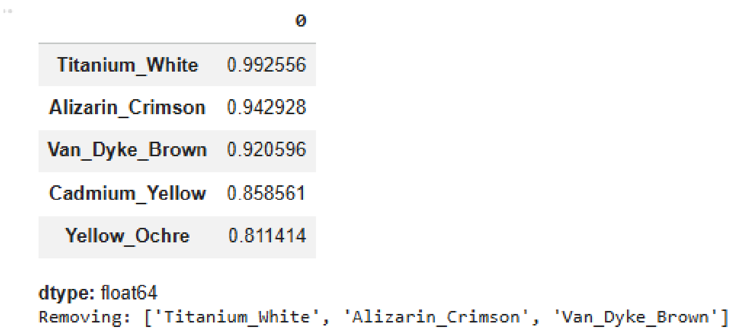
Figure 3: Diagnosis of Document Frequency Distribution (DF=1 Proportion)



```
from  nltk.tokenize  import  word_tokenize
from  sklearn.feature_extraction.text  import  CountVectorizer

#model
cv_diag=  CountVectorizer(stop_words='english',  lowercase=True)  #  max_df=  0.95,  m

#  fit
cv_diag.fit(title_df["title"])


X  =  cv_diag.transform(title_df['title'])
df_counts  =  np.asarray((X  >  0).sum(axis=0)).ravel()

words  =  cv_diag.get_feature_names_out()
df_table  =  pd.DataFrame({
        "word":  words,
        "df":  df_counts
}).sort_values("df")

num_df1  =  (df_table["df"]  ==  1).sum()
pct_df1  =  num_df1  /  len(df_table)

print("Total  vocab:",  len(df_table))
print("DF=1  words:",  num_df1,  f"({pct_df1:.2%})")
```

```
Total vocab: 328
DF=1 words: 189 (57.62%)
```

Figure 4: High-Frequency Paint Colors and Feature Filtering

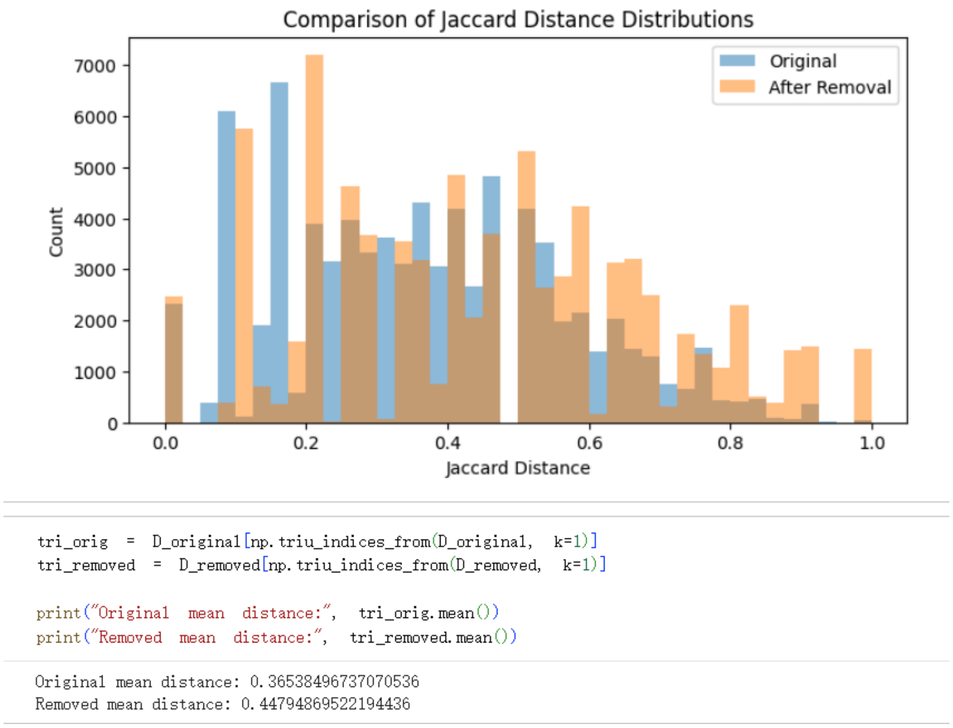


	0
Titanium_White	0.992556
Alizarin_Crimson	0.942928
Van_Dyke_Brown	0.920596
Cadmium_Yellow	0.858561
Yellow_Ochre	0.811414

dtype: float64

```
Removing: ['Titanium_White', 'Alizarin_Crimson', 'Van_Dyke_Brown']
```

Figure 5: Comparison of Jaccard Distance Distributions and Mean Distribution



```
tri_orig = D_original[np.triu_indices_from(D_original, k=1)]
tri_removed = D_removed[np.triu_indices_from(D_removed, k=1)]

print(“Original mean distance:”, tri_orig.mean())
print(“Removed mean distance:”, tri_removed.mean())
```

```
Original mean distance: 0.36538496737070536
Removed mean distance: 0.44794869522194436
```

Figure 6: DBSCAN K-distance plot (min_support=5)

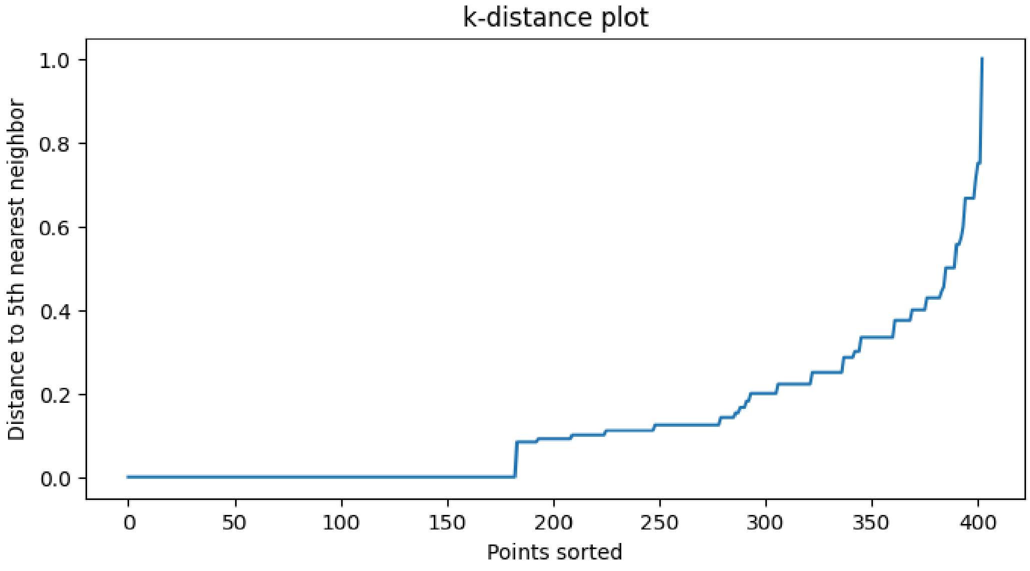
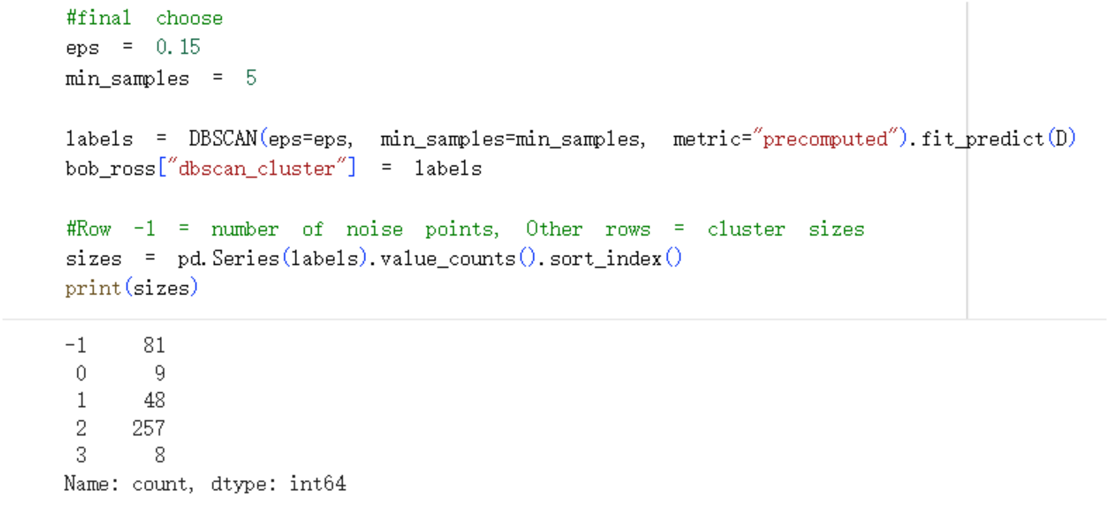


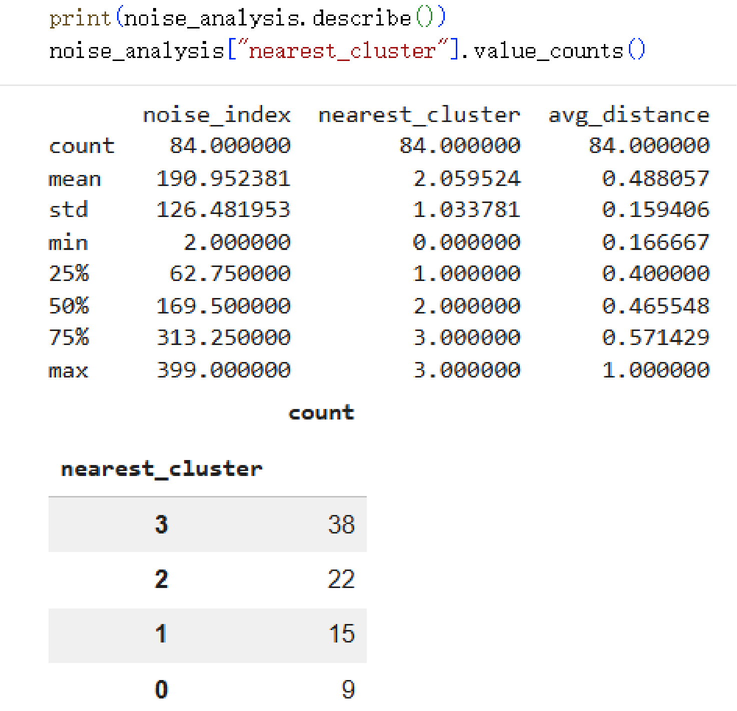
Figure 7: Final DBSCAN Parameters and Cluster Size Distribution



```
#final  choose
eps  =  0.15
min_samples  =  5

labels  =  DBSCAN(eps=eps,  min_samples=min_samples,  metric="precomputed").fit_predict(D)
bob_ross["dbscan_cluster"]  =  labels

#Row  -1  =  number  of  noise  points,  Other  rows  =  cluster  sizes
sizes  =  pd.Series(labels).value_counts().sort_index()
print(sizes)
```

```
-1     81
 0      9
 1     48
 2    257
 3      8
Name: count, dtype: int64
```

Figure 8: DBSCAN Noise Analysis and Nearest Cluster



```
print(noise_analysis.describe())
noise_analysis["nearest_cluster"].value_counts()
```

```
       noise_index  nearest_cluster  avg_distance
count    84.000000        84.000000     84.000000
mean    190.952381         2.059524      0.488057
std     126.481953         1.033781      0.159406
min       2.000000         0.000000      0.166667
25%      62.750000         1.000000      0.400000
50%     169.500000         2.000000      0.465548
75%     313.250000         3.000000      0.571429
max     399.000000         3.000000      1.000000
```

	count
nearest_cluster	
3	38
2	22
1	15
0	9

Figure 9: BoW Cosine Similarity Between Palette Clusters (Title)

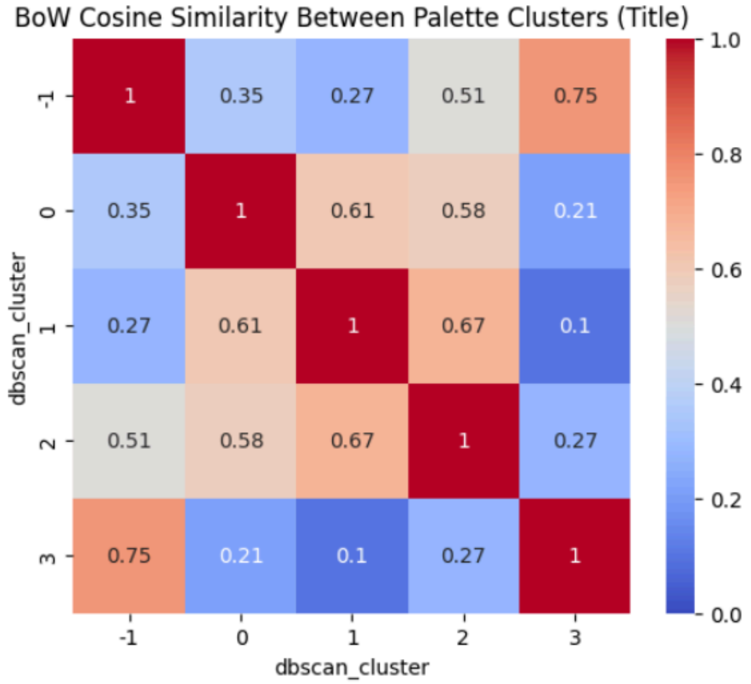


Figure 10: DBSCAN Cluster-Level Paint Emphasis (Lift)

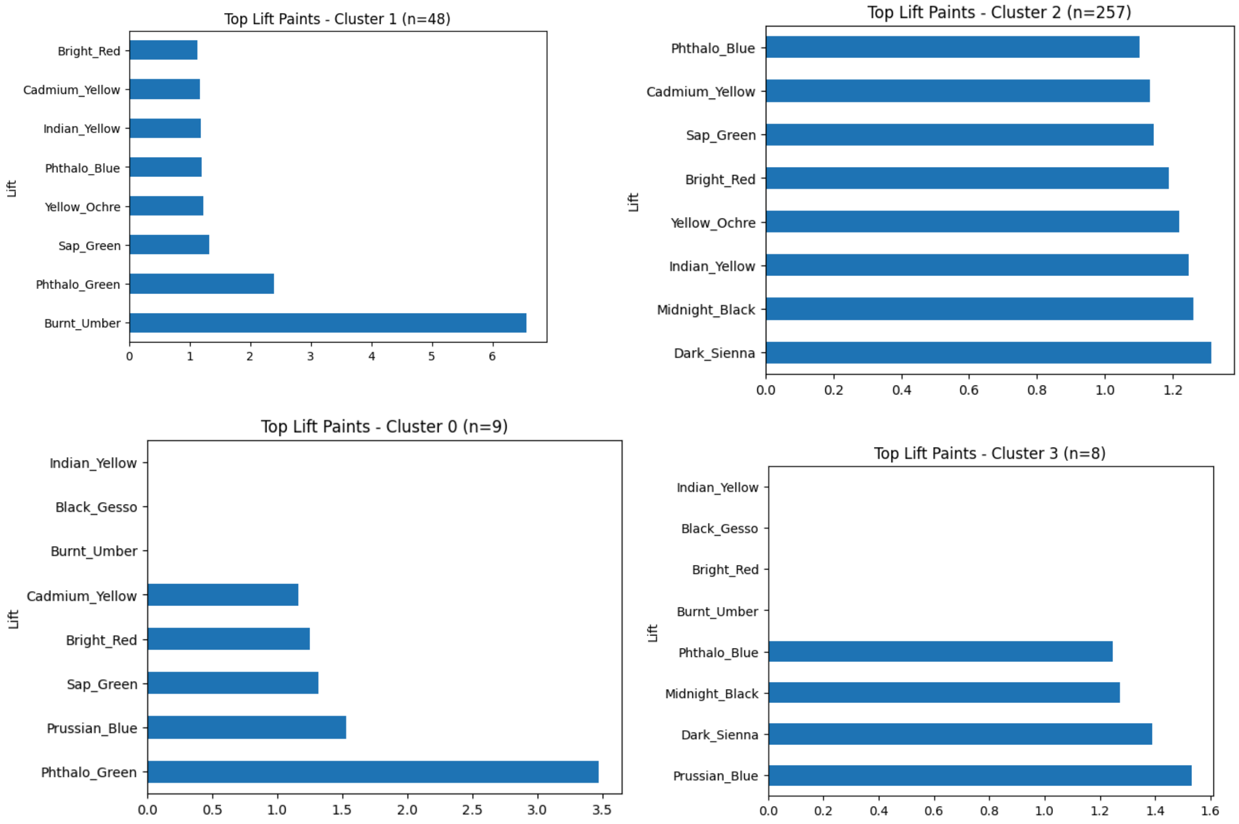
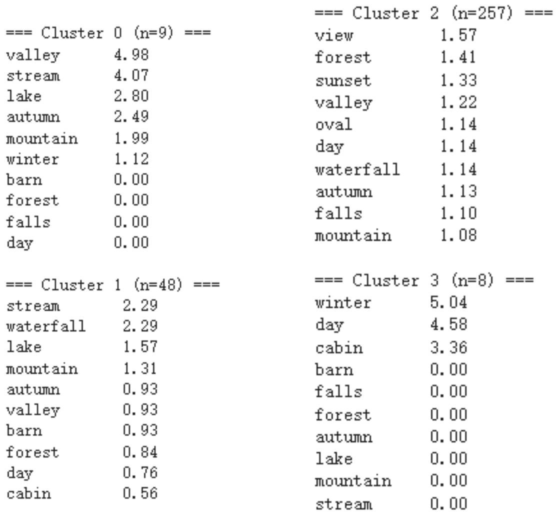


Figure 10: BowW Cluster-Level Keyword Lift in Title



```
=== Cluster 0 (n=9) ===
valley      4.98
stream      4.07
lake        2.80
autumn      2.49
mountain    1.99
winter      1.12
barn        0.00
forest      0.00
falls       0.00
day         0.00

=== Cluster 1 (n=48) ===
stream       2.29
waterfall    2.29
lake         1.57
mountain     1.31
autumn       0.93
valley       0.93
barn         0.93
forest       0.84
day          0.76
cabin        0.56

=== Cluster 2 (n=257) ===
view         1.57
forest       1.41
sunset       1.33
valley       1.22
oval         1.14
day          1.14
waterfall    1.14
autumn       1.13
falls        1.10
mountain     1.08

=== Cluster 3 (n=8) ===
winter      5.04
day         4.58
cabin       3.36
barn        0.00
falls       0.00
forest      0.00
autumn      0.00
lake        0.00
mountain    0.00
stream      0.00
```

Table1: Summary of Cluster Characteristics: Color Emphasis, Themes, and Interpretation

Cluster	Color emphasis	Theme keywords	Interpretation
Cluster 0 (n = 9)	Phthalo Green Sap Green Bright Red Cadmium Yellow	valley stream lake autumn mountain	Green-dominant landscape scenes, often centered on water (streams, lakes) and open valleys.
Cluster 1 (n = 48)	Burnt Umber Phthalo Green Yellow Ochre Cadmium Yellow Bright Red	stream waterfall lake mountain autumn	Earth-tone–heavy compositions with natural water elements. Warmer, light-enhanced landscape

Cluster 2 (n = 257)	Balanced usage across many common paints	view forest sunset valley waterfall autumn	Baseline Bob Ross palette. General landscape themes without strong specialization.
Cluster 3 (n = 8)	Prussian Blue Dark Sienna Midnight Black Phthalo Blue	winter cabin	Darker, blue-heavy winter scenes, often featuring cabins.

Process Overview

1. Data loading: Load the Bob Ross dataset (403 paintings) and select the 18 paint indicator columns.
2. Data validation: Confirm paint indicators are binary, with no missing values/duplicates in the feature matrix.
3. EDA:
    a. Summarize global paint usage rates and assess palette similarity using pairwise Jaccard similarity to confirm recurring palette templates.
    b. Feature setup: Use the 18 paint indicators as binary vectors; treat each painting's palette as a set of paints.
4. Method 1: Hierarchical Clustering
5. Method 2: Association rules (global & within-cluster)
6. **DBSCAN:**
    a. Distance preparation: Compute pairwise Jaccard distance between paintings based on binary paint usage.
    b. Feature refinement: Remove near-universal colors to improve sensitivity and avoid similarity compression.
    c. Parameter selection:Use a k-distance plot to identify a reasonable eps range and perform grid search over eps and min_samples.
    d. Final clustering:Select final parameters based on cluster count, noise rate, and silhouette score.
    e. Cluster interpretation: Interpret clusters using paint-level lift and analyze noise points to assess boundary vs. isolated variations.

    f. Noise analysis: measuring their average distance to each cluster to determine whether they represent isolated styles or boundary variations.

7. **BoW:**
    a. Text preprocessing: Preprocess painting titles (lowercasing, stopword removal)
    b. Feature representation: Represent titles using binary Bag-of-Words
    c. Vocabulary diagnosis: Diagnose document frequency distribution and apply min_df to reduce noise.
    d. Cluster-level keyword analysis: Compute word-level lift within each DBSCAN cluster and apply a global frequency filter (min_global) to stabilize results.
    e. Thematic similarity assessment: Measure thematic similarity across clusters using cosine similarity and compare title themes with color-based clusters.

Use of Generative AI Tools

Link: https://chatgpt.com/share/699bf14a-3114-8003-a663-f90e89d5afe5

I used ChatGPT as a learning and coding-support tool for this milestone because I was not familiar with how DBSCAN works or how to apply it appropriately. I also asked for suggestions when applying BoW, as well as for refining and grammar-checking my writing.

I first asked ChatGPT to explain how DBSCAN works conceptually, including what density-based clustering is, the role of eps and min_samples, how noise points are defined, and how it aligns with Jaccard distance. Next, I asked for coding guidance for DBSCAN. After understanding the concept, I asked step-by-step guidance to implement the DBSCAN pipeline in Python. This included building and interpreting a k-distance plot (testing different min_samples values such as 3, 5, and 8) to identify a reasonable eps search range, writing the main DBSCAN code using either metric="jaccard" directly or metric="precomputed" with a distance matrix, and summarizing outputs such as the number of clusters, cluster sizes, and noise count.

Additionally, I asked whether removing extremely high-frequency paints could improve clustering sensitivity. ChatGPT explained that near-universal colors can inflate similarity under Jaccard and compress the distance space, potentially masking meaningful structure. Based on this reasoning, I created my own diagnostic to compare pairwise similarity/distance distributions before and after removing the top three globally dominant colors.

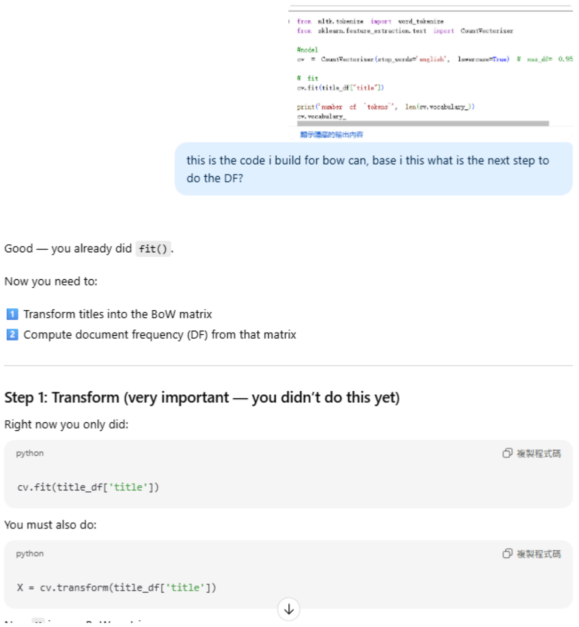
Finally, for DBSCAN, I asked how to structure a parameter tuning table and how to interpret metrics such as noise rate and silhouette score. ChatGPT suggested using reasonable constraints to avoid degenerate solutions such as too many micro-clusters or excessive noise. I then implemented a tuning grid and applied my own filtering rules. After the final choice, I asked how to do the lift table for understanding the color emphasized in each cluster, from step by step code and explanations, the code was using 'for loop' so I asked for the purpose and asked step by step to understand how the code works.

For the BoW analysis, after completing the basic coding part, I asked whether there were any refinements I should consider. I specifically asked how to define min_df and how to check whether many words appear in only one painting title. ChatGPT suggested examining the document frequency distribution and quantifying how many words appear only once. Following this guidance, I created my own diagnostic showing that over 57% of words appear in only one title. Based on that result, I chose min_df = 2 to reduce noise and stabilize cluster-level comparisons.

I also asked whether computing lift was appropriate. I asked how to define min_global and how to interpret large lift values. ChatGPT explained that extremely low base rates can inflate lift and suggested applying a minimum global frequency threshold. I tested min_global = 0.01 and 0.02 and selected 0.02 because it reduced extreme lift spikes and produced more interpretable results.
Finally, after organizing the cluster summaries for both DBSCAN (color usage) and BoW (title keywords), I asked whether my interpretations were reasonable and suggestions for improvement. Additionally, ask for grammar and spelling errors checks for writing.

BoW define min_df and coding guidance for DF distribution

Here is the code structure i used:



```
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import CountVectorizer

#model
cv = CountVectorizer(stop_words='english', lowercase=True) # max_df= 0.95

# fit
cv.fit(title_df['title'])

print('number of `tokens`', len(cv.vocabulary_))
cv.vocabulary_
```

this is the code i build for bow can, base i this what is the next step to do the DF?

Good — you already did `fit()`.

Now you need to:

1️⃣ Transform titles into the BoW matrix
2️⃣ Compute document frequency (DF) from that matrix

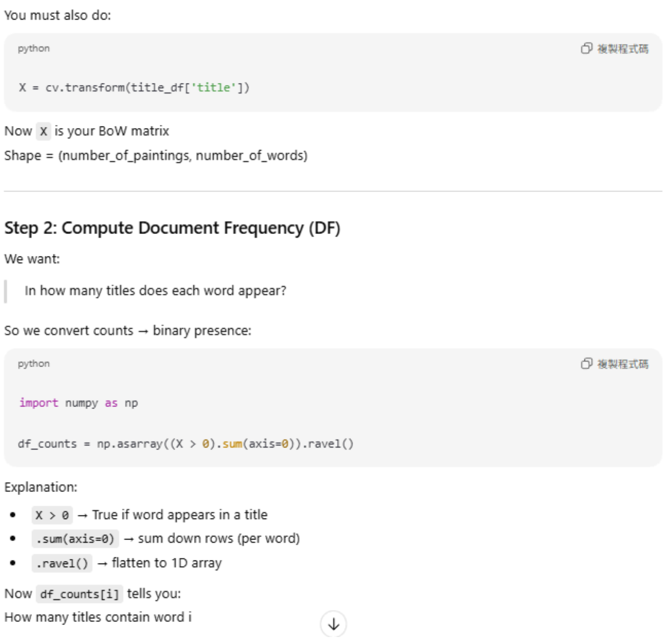
Step 1: Transform (very important — you didn't do this yet)

Right now you only did:

```python
cv.fit(title_df['title'])
```

You must also do:

```python
X = cv.transform(title_df['title'])
```



You must also do:

```python
X = cv.transform(title_df['title'])
```

Now `X` is your BoW matrix
Shape = (number_of_paintings, number_of_words)

Step 2: Compute Document Frequency (DF)

We want:

> In how many titles does each word appear?

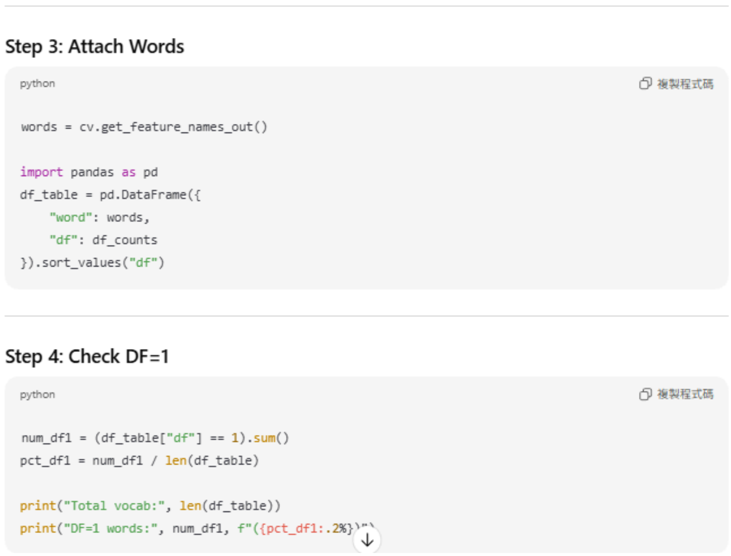
So we convert counts → binary presence:

```python
import numpy as np

df_counts = np.asarray((X > 0).sum(axis=0)).ravel()
```

Explanation:

- `X > 0` → True if word appears in a title
- `.sum(axis=0)` → sum down rows (per word)
- `.ravel()` → flatten to 1D array

Now `df_counts[i]` tells you:
How many titles contain word i



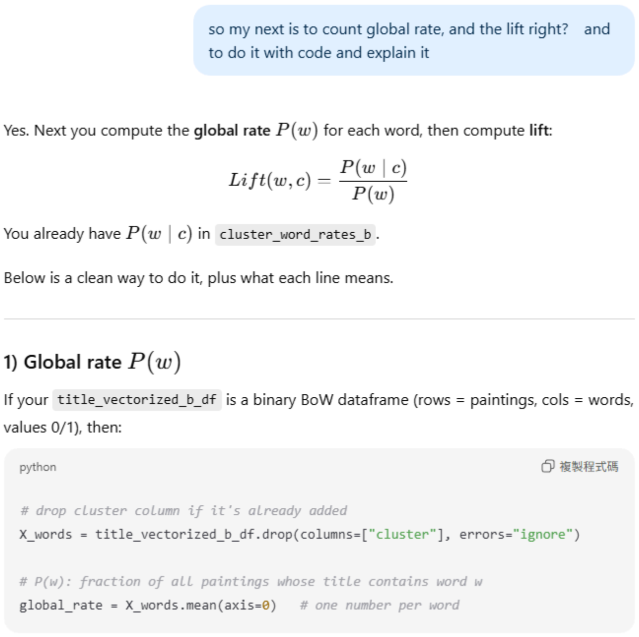
Step 3: Attach Words

```python
words = cv.get_feature_names_out()

import pandas as pd
df_table = pd.DataFrame({
    "word": words,
    "df": df_counts
}).sort_values("df")
```

Step 4: Check DF=1

```python
num_df1 = (df_table["df"] == 1).sum()
pct_df1 = num_df1 / len(df_table)

print("Total vocab:", len(df_table))
print("DF=1 words:", num_df1, f"({pct_df1:.2%})")
```

BoW Lift table and Min_global explanation



so my next is to count global rate, and the lift right? and to do it with code and explain it

Yes. Next you compute the **global rate** $P(w)$ for each word, then compute **lift**:

$$Lift(w, c) = \frac{P(w \mid c)}{P(w)}$$

You already have $P(w \mid c)$ in `cluster_word_rates_b`.

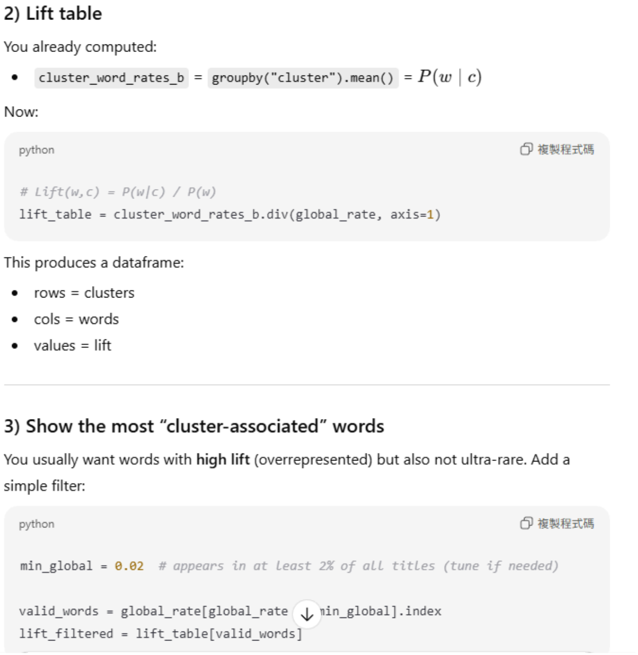
Below is a clean way to do it, plus what each line means.

---

1) Global rate $P(w)$

If your `title_vectorized_b_df` is a binary BoW dataframe (rows = paintings, cols = words, values 0/1), then:

```python
# drop cluster column if it's already added
X_words = title_vectorized_b_df.drop(columns=["cluster"], errors="ignore")

# P(w): fraction of all paintings whose title contains word w
global_rate = X_words.mean(axis=0)   # one number per word
```



2) Lift table

You already computed:

- `cluster_word_rates_b` = `groupby("cluster").mean()` = $P(w \mid c)$

Now:

```python
# Lift(w,c) = P(w|c) / P(w)
lift_table = cluster_word_rates_b.div(global_rate, axis=1)
```

This produces a dataframe:

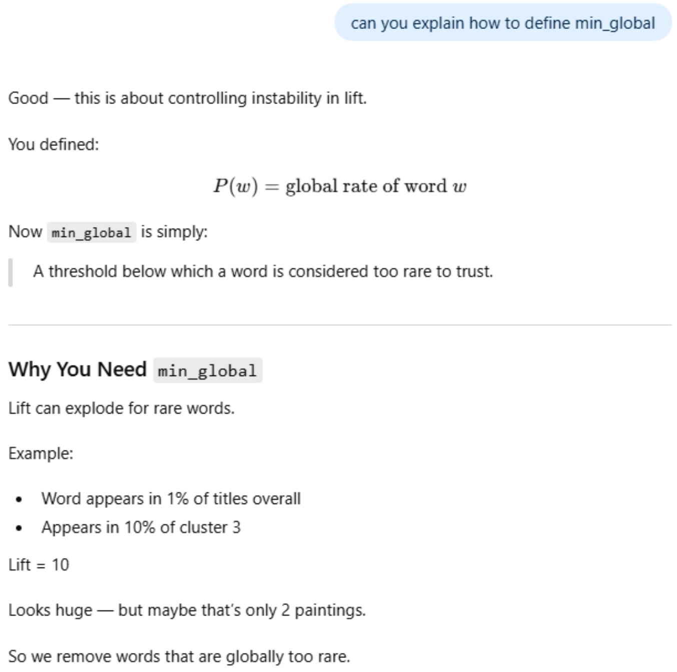
- rows = clusters
- cols = words
- values = lift

---

3) Show the most “cluster-associated” words

You usually want words with **high lift** (overrepresented) but also not ultra-rare. Add a simple filter:

```python
min_global = 0.02  # appears in at least 2% of all titles (tune if needed)

valid_words = global_rate[global_rate  nin_global].index
lift_filtered = lift_table[valid_words]
```



can you explain how to define min_global

Good — this is about controlling instability in lift.

You defined:

$$P(w) = \text{global rate of word } w$$

Now `min_global` is simply:

> A threshold below which a word is considered too rare to trust.

Why You Need `min_global`

Lift can explode for rare words.

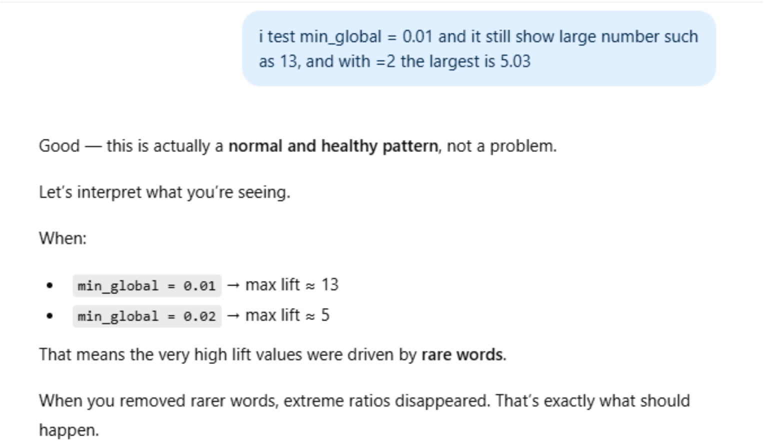
Example:

- Word appears in 1% of titles overall
- Appears in 10% of cluster 3

Lift = 10

Looks huge — but maybe that's only 2 paintings.

So we remove words that are globally too rare.



i test min_global = 0.01 and it still show large number such as 13, and with =2 the largest is 5.03

Good — this is actually a **normal and healthy pattern**, not a problem.

Let's interpret what you're seeing.

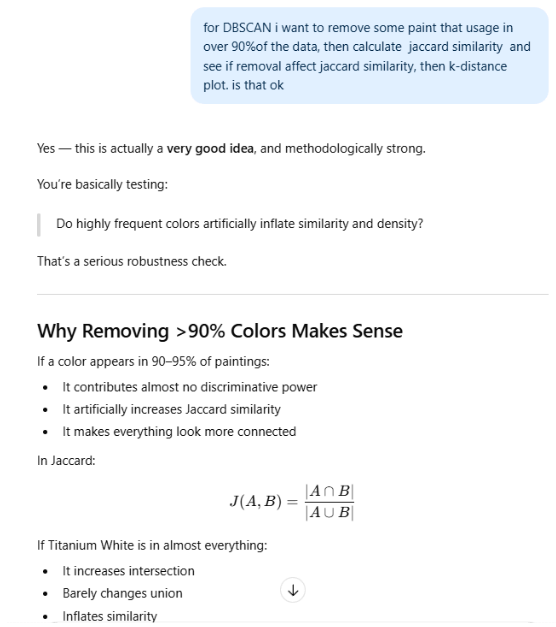
When:

- `min_global = 0.01` → max lift ≈ 13
- `min_global = 0.02` → max lift ≈ 5

That means the very high lift values were driven by **rare words**.

When you removed rarer words, extreme ratios disappeared. That's exactly what should happen.

DBSCAN conceptually explain and coding guidance
Distance Histogram/ k-distance plot/ DBSCAN pipeline



for DBSCAN i want to remove some paint that usage in over 90%of the data, then calculate jaccard similarity and see if removal affect jaccard similarity, then k-distance plot. is that ok

Yes — this is actually a **very good idea**, and methodologically strong.

You're basically testing:

> Do highly frequent colors artificially inflate similarity and density?

That's a serious robustness check.

Why Removing >90% Colors Makes Sense

If a color appears in 90–95% of paintings:

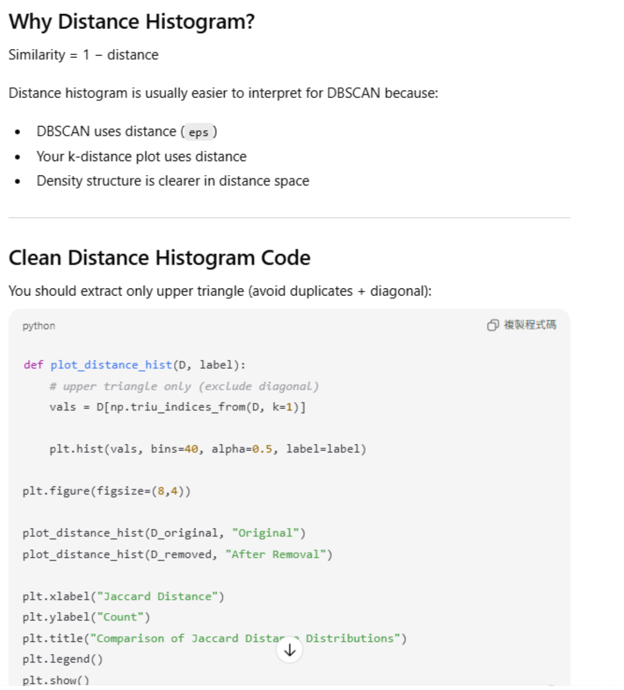
- It contributes almost no discriminative power
- It artificially increases Jaccard similarity
- It makes everything look more connected

In Jaccard:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

If Titanium White is in almost everything:

- It increases intersection
- Barely changes union
- Inflates similarity



Why Distance Histogram?

Similarity = 1 − distance

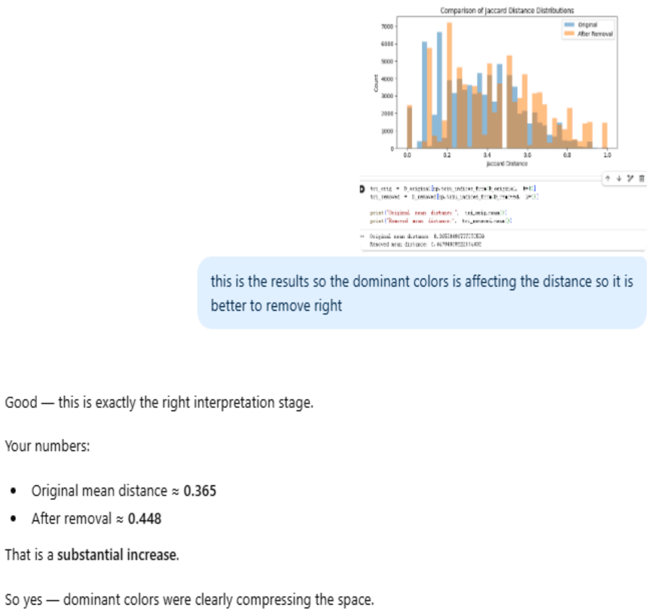
Distance histogram is usually easier to interpret for DBSCAN because:

- DBSCAN uses distance (`eps`)
- Your k-distance plot uses distance
- Density structure is clearer in distance space

Clean Distance Histogram Code

You should extract only upper triangle (avoid duplicates + diagonal):

```python
def plot_distance_hist(D, label):
    # upper triangle only (exclude diagonal)
    vals = D[np.triu_indices_from(D, k=1)]

    plt.hist(vals, bins=40, alpha=0.5, label=label)

plt.figure(figsize=(8,4))

plot_distance_hist(D_original, "Original")
plot_distance_hist(D_removed, "After Removal")

plt.xlabel("Jaccard Distance")
plt.ylabel("Count")
plt.title("Comparison of Jaccard Distar Distributions")
plt.legend()
plt.show()
```



this is the results so the dominant colors is affecting the distance so it is better to remove right

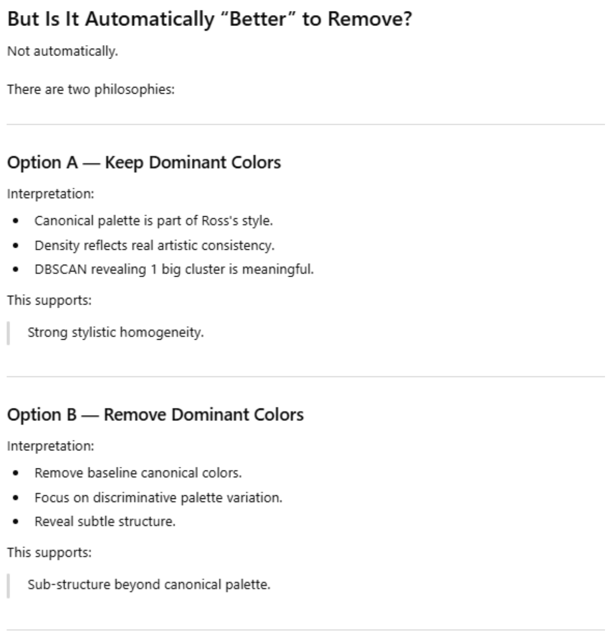
Good — this is exactly the right interpretation stage.

Your numbers:

- Original mean distance ≈ **0.365**
- After removal ≈ **0.448**

That is a **substantial increase.**

So yes — dominant colors were clearly compressing the space.



But Is It Automatically "Better" to Remove?

Not automatically.

There are two philosophies:

Option A — Keep Dominant Colors

Interpretation:

- Canonical palette is part of Ross's style.
- Density reflects real artistic consistency.
- DBSCAN revealing 1 big cluster is meaningful.

This supports:

> Strong stylistic homogeneity.

Option B — Remove Dominant Colors

Interpretation:

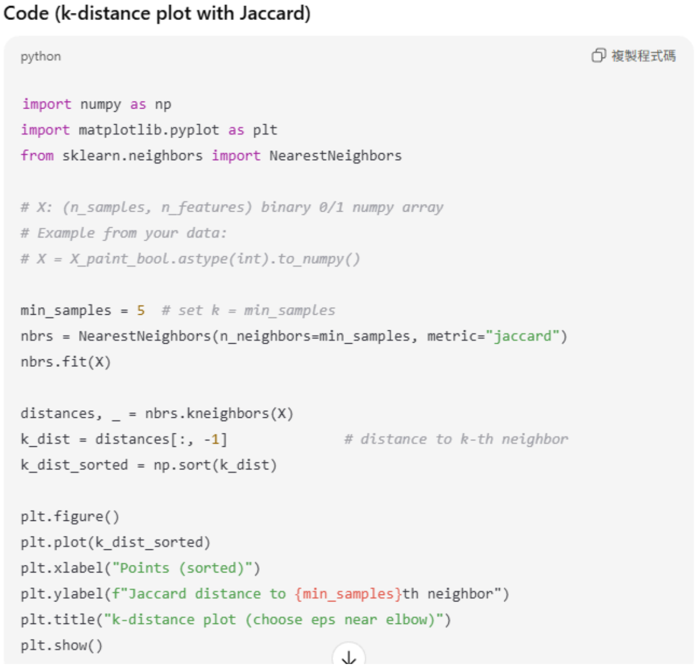
- Remove baseline canonical colors.
- Focus on discriminative palette variation.
- Reveal subtle structure.

This supports:

> Sub-structure beyond canonical palette.

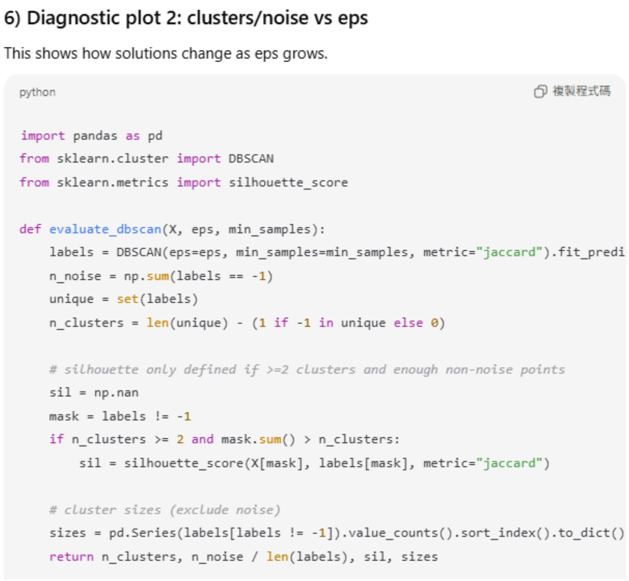
DBSCAN conceptually explain and coding guidance

Here is the code structure i used:



Code (k-distance plot with Jaccard)

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import NearestNeighbors

# X: (n_samples, n_features) binary 0/1 numpy array
# Example from your data:
# X = X_paint_bool.astype(int).to_numpy()

min_samples = 5  # set k = min_samples
nbrs = NearestNeighbors(n_neighbors=min_samples, metric="jaccard")
nbrs.fit(X)

distances, _ = nbrs.kneighbors(X)
k_dist = distances[:, -1]              # distance to k-th neighbor
k_dist_sorted = np.sort(k_dist)

plt.figure()
plt.plot(k_dist_sorted)
plt.xlabel("Points (sorted)")
plt.ylabel(f"Jaccard distance to {min_samples}th neighbor")
plt.title("k-distance plot (choose eps near elbow)")
plt.show()
```



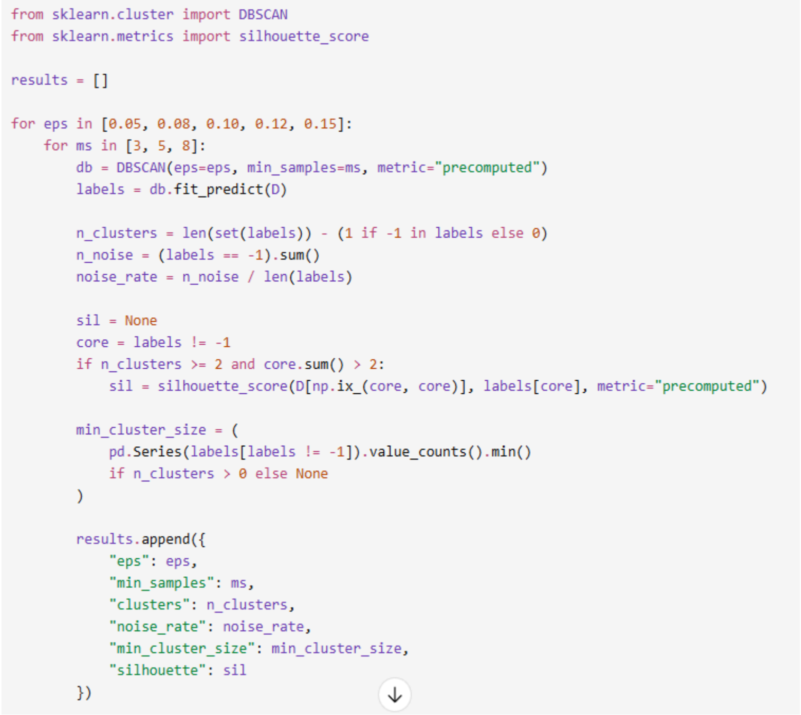
6) Diagnostic plot 2: clusters/noise vs eps

This shows how solutions change as eps grows.

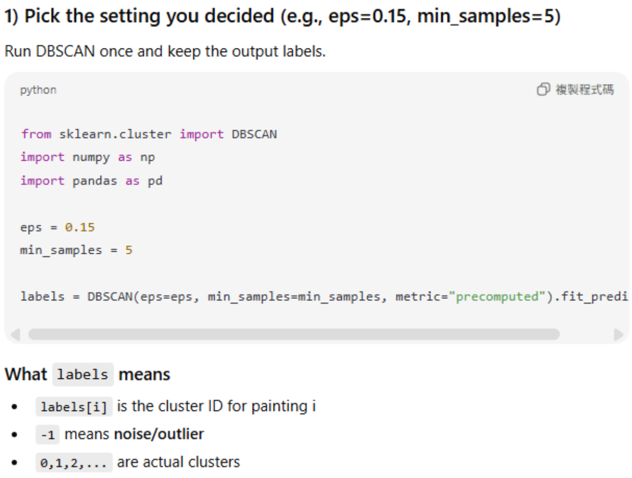
```python
import pandas as pd
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score

def evaluate_dbscan(X, eps, min_samples):
    labels = DBSCAN(eps=eps, min_samples=min_samples, metric="jaccard").fit_predi
    n_noise = np.sum(labels == -1)
    unique = set(labels)
    n_clusters = len(unique) - (1 if -1 in unique else 0)

    # silhouette only defined if >=2 clusters and enough non-noise points
    sil = np.nan
    mask = labels != -1
    if n_clusters >= 2 and mask.sum() > n_clusters:
        sil = silhouette_score(X[mask], labels[mask], metric="jaccard")

    # cluster sizes (exclude noise)
    sizes = pd.Series(labels[labels != -1]).value_counts().sort_index().to_dict()
    return n_clusters, n_noise / len(labels), sil, sizes
```

DBSCAN Pipeline coding guidence

Here is the code structure i used:



```python
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score

results = []

for eps in [0.05, 0.08, 0.10, 0.12, 0.15]:
    for ms in [3, 5, 8]:
        db = DBSCAN(eps=eps, min_samples=ms, metric="precomputed")
        labels = db.fit_predict(D)

        n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
        n_noise = (labels == -1).sum()
        noise_rate = n_noise / len(labels)

        sil = None
        core = labels != -1
        if n_clusters >= 2 and core.sum() > 2:
            sil = silhouette_score(D[np.ix_(core, core)], labels[core], metric="precomputed")

        min_cluster_size = (
            pd.Series(labels[labels != -1]).value_counts().min()
            if n_clusters > 0 else None
        )

        results.append({
            "eps": eps,
            "min_samples": ms,
            "clusters": n_clusters,
            "noise_rate": noise_rate,
            "min_cluster_size": min_cluster_size,
            "silhouette": sil
        })
```



1) Pick the setting you decided (e.g., eps=0.15, min_samples=5)

Run DBSCAN once and keep the output labels.

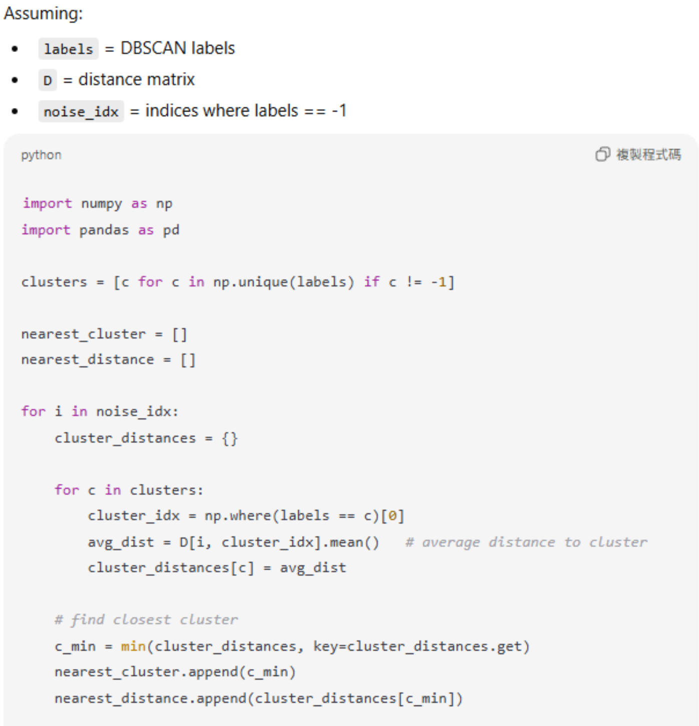
```python
from sklearn.cluster import DBSCAN
import numpy as np
import pandas as pd

eps = 0.15
min_samples = 5

labels = DBSCAN(eps=eps, min_samples=min_samples, metric="precomputed").fit_predi
```

What `labels` means

- `labels[i]` is the cluster ID for painting i
- `-1` means **noise/outlier**
- `0,1,2,...` are actual clusters

DBSCAN Noise points asking for distance to nearest cluster for understanding

Here is the code structure i used:



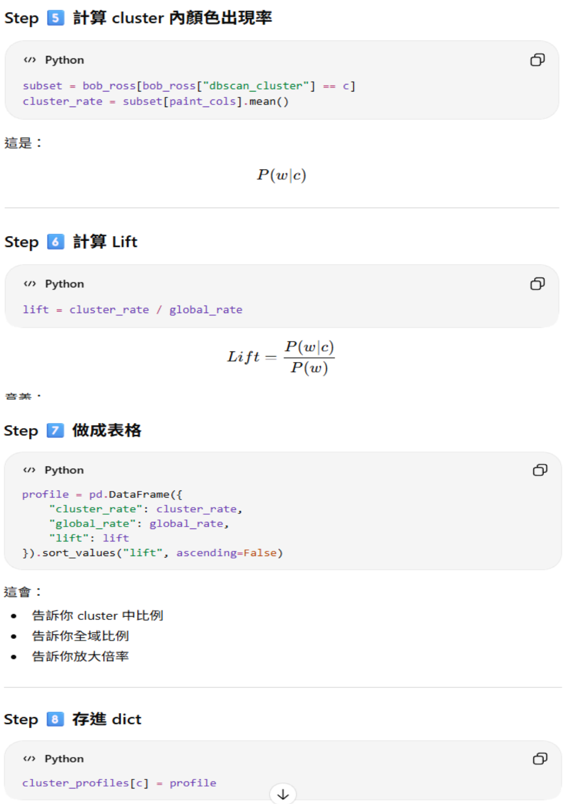
Assuming:

- `labels` = DBSCAN labels
- `D` = distance matrix
- `noise_idx` = indices where labels == -1

```python
import numpy as np
import pandas as pd

clusters = [c for c in np.unique(labels) if c != -1]

nearest_cluster = []
nearest_distance = []

for i in noise_idx:
    cluster_distances = {}

    for c in clusters:
        cluster_idx = np.where(labels == c)[0]
        avg_dist = D[i, cluster_idx].mean()    # average distance to cluster
        cluster_distances[c] = avg_dist

    # find closest cluster
    c_min = min(cluster_distances, key=cluster_distances.get)
    nearest_cluster.append(c_min)
    nearest_distance.append(cluster_distances[c_min])
```

DBSCAN Lift table compute coding guidance and explanations

Here is the code structure i used:



Step 5 計算 cluster 內顏色出現率

```python
subset = bob_ross[bob_ross["dbscan_cluster"] == c]
cluster_rate = subset[paint_cols].mean()
```

這是：

$$P(w|c)$$

Step 6 計算 Lift

```python
lift = cluster_rate / global_rate
```

$$Lift = \frac{P(w|c)}{P(w)}$$

意義：

Step 7 做成表格

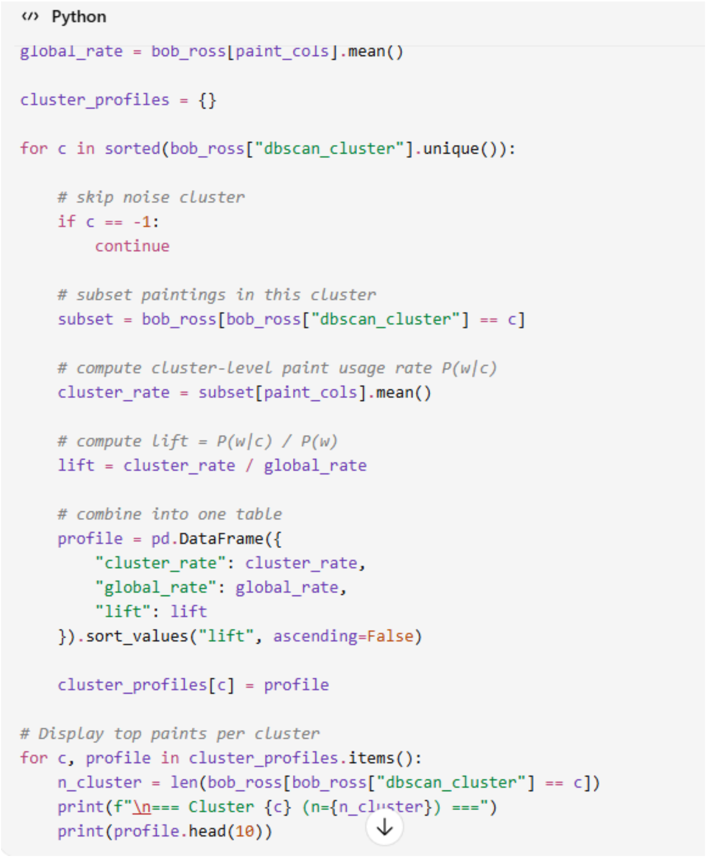
```python
profile = pd.DataFrame({
    "cluster_rate": cluster_rate,
    "global_rate": global_rate,
    "lift": lift
}).sort_values("lift", ascending=False)
```

這會：

- 告訴你 cluster 中比例
- 告訴你全域比例
- 告訴你放大倍率

Step 8 存進 dict

```python
cluster_profiles[c] = profile
```



```python
global_rate = bob_ross[paint_cols].mean()

cluster_profiles = {}

for c in sorted(bob_ross["dbscan_cluster"].unique()):

    # skip noise cluster
    if c == -1:
        continue

    # subset paintings in this cluster
    subset = bob_ross[bob_ross["dbscan_cluster"] == c]

    # compute cluster-level paint usage rate P(w|c)
    cluster_rate = subset[paint_cols].mean()

    # compute lift = P(w|c) / P(w)
    lift = cluster_rate / global_rate

    # combine into one table
    profile = pd.DataFrame({
        "cluster_rate": cluster_rate,
        "global_rate": global_rate,
        "lift": lift
    }).sort_values("lift", ascending=False)

    cluster_profiles[c] = profile

# Display top paints per cluster
for c, profile in cluster_profiles.items():
    n_cluster = len(bob_ross[bob_ross["dbscan_cluster"] == c])
    print(f"\n=== Cluster {c} (n={n_cluster}) ===")
    print(profile.head(10))
```