Computational Intelligence and Decision Making - Lab Work 2

Full Name: Ravinthiran Partheepan | Degree: Master's in Aritificial Intelligence | Semester: 1st / Year: 1st

Problem Statement:

Feature extraction and results analysis using Unsupervised Learning

Problem: apply subject classification based on features extracted using unsupervised learning algorithm (k-means) from image. Perform result analysis and provide the insights by using Self-Organizing-Maps (SOM).

Task 1 – Film posters.

Task 2 – Music album covers.

Task 3 – Book covers.

Task 4 – Points of Interest.

Project workflow:

- Collect data: image(-s) and main category
- Implement algorithm to extract features from image(-s) using k-means (k represents the number of color vectors in rgb values). In the feature vector, the color vectors are sorted from the ones representing the largest cluster to the smallest.
- Use KNN algorithm to evaluate the extracted features using the main category as label. Based on the results, select the appropriate number of clusters for feature extraction.
- Implement SOM for the dataset using extracted features as an input vector. Analyze the results (provide grids with colors representing the most frequent category of the node, the dominant color for the node, ...). Comment on the results whether the pattern between categories is logical, the similarity makes sense, etc.
  ### Libraries used: Sklearn, Matplotlib, Seaborn, and VertezML

Note: I have used vertezML library for experimentation purpose which will be helpful for me to compare with other frameworks like sklearn in terms of performance, how it handles massive computation process, and most importantly it will help me to identify backlogs and improve it.

- Github Repository: https://github.com/ravinthiranpartheepan1407/vertez
- Py Pakage: https://pypi.org/project/vertezml/
- Documentation: www.vertez.org (In Progress)

Feel Free to leave any suggestions or feedback if you have any related to any ML math metric!

Mini Project Github Repository Link: https://github.com/ravinthiranpartheepan1407/self-organizing-maps

Libraries Import

In [1]:
```
import vertezml as vz
import numpy as np
import matplotlib.pyplot as plt
from bs4 import BeautifulSoup
import requests
import pandas as pd
import os
import cv2
from sklearn.cluster import KMeans
from sklearn.neighbors import KNeighborsClassifier
from minisom import MiniSom
import seaborn as sns
from PIL import Image
from IPython.display import display
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from scipy.stats import mode
```

Data Import

In [99]:
```
# Scarping Marvel Film Posters
wikipedia_url = "https://en.wikipedia.org/wiki/Category:Marvel_Cinematic_Universe_film_p

# Created a folder to save the downloaded posters
output_folder = "marvel_comics_posters"
os.makedirs(output_folder, exist_ok=True)

# GET request to the Wikipedia page
response = requests.get(wikipedia_url)

def download_image(url, save_path):
    try:
        response = requests.get(url)

        # Check if the request was successful
        if response.status_code == 200:
            with open(save_path, 'wb') as file:
                file.write(response.content)
            print(f"Image downloaded and saved to {save_path}")
        else:
            print(f"Failed to download the image. Status code: {response.status_code}")

    except Exception as e:
        print(f"An error occurred: {str(e)}")

# Checking if the request was successful or not
if response.status_code == 200:
    # Parsing the HTML content of the page
    soup = BeautifulSoup(response.text, 'html.parser')

    # Finding all href elements on the page
    href_elements = soup.find_all('a')

    # Looping through the href elements and downloading images from the linked pages
    for href in href_elements:
        img_url = href.get('href')
        if img_url and img_url.startswith("/wiki/File:"):
            img_page_url = "https://en.wikipedia.org" + img_url
            img_page_response = requests.get(img_page_url)
            img_page_soup = BeautifulSoup(img_page_response.text, 'html.parser')
```

```
            img_element = img_page_soup.find('div', {'class': 'fullImageLink'})
            if img_element:
                img_link = img_element.find('a')['href']
                image_url = "https:" + img_link
                save_path = os.path.join(output_folder, os.path.basename(image_url))
                download_image(image_url, save_path)

print("Marvel Film posters downloaded.")
```

```
Image downloaded and saved to marvel_comics_posters\Ant-Man_%28film%29_poster.jpg
Failed to download the image. Status code: 403
Image downloaded and saved to marvel_comics_posters\Ant-Man_and_the_Wasp_Quantumania_pos
ter.jpg
Failed to download the image. Status code: 403
Image downloaded and saved to marvel_comics_posters\The_Avengers_%282012_film%29_poster.
jpg
Image downloaded and saved to marvel_comics_posters\Avengers_Age_of_Ultron_poster.jpg
Image downloaded and saved to marvel_comics_posters\Avengers_Endgame_poster.jpg
Failed to download the image. Status code: 403
Image downloaded and saved to marvel_comics_posters\Avengers_Infinity_War_poster.jpg
Image downloaded and saved to marvel_comics_posters\Black_Panther_%28film%29_poster.jpg
Image downloaded and saved to marvel_comics_posters\Black_Panther_Wakanda_Forever_poste
r.jpg
Image downloaded and saved to marvel_comics_posters\Black_Widow_%282021_film%29_poster.j
pg
Image downloaded and saved to marvel_comics_posters\Captain_America_Brave_New_World_log
o.jpg
Image downloaded and saved to marvel_comics_posters\Captain_America_Civil_War_poster.jpg
Image downloaded and saved to marvel_comics_posters\Captain_America_The_First_Avenger_po
ster.jpg
Image downloaded and saved to marvel_comics_posters\Captain_America_The_Winter_Soldier_p
oster.jpg
Image downloaded and saved to marvel_comics_posters\Captain_Marvel_%28film%29_poster.jpg
Failed to download the image. Status code: 403
Image downloaded and saved to marvel_comics_posters\Deadpool_3_logo.jpg
Image downloaded and saved to marvel_comics_posters\Doctor_Strange_%282016_film%29_poste
r.jpg
Image downloaded and saved to marvel_comics_posters\Doctor_Strange_in_the_Multiverse_of_
Madness_poster.jpg
Image downloaded and saved to marvel_comics_posters\Emily_VanCamp_as_Sharon_Carter_in_Ca
ptain_America_The_Winter_Soldier_poster.jpg
Image downloaded and saved to marvel_comics_posters\Eternals_%28film%29_poster.jpeg
Image downloaded and saved to marvel_comics_posters\Guardians_of_the_Galaxy_%28film%29_p
oster.jpg
Image downloaded and saved to marvel_comics_posters\Guardians_of_the_Galaxy_Vol._2_poste
r.jpg
Image downloaded and saved to marvel_comics_posters\Guardians_of_the_Galaxy_Vol._3_poste
r.jpg
Image downloaded and saved to marvel_comics_posters\The_Incredible_Hulk_%28film%29_poste
r.jpg
Image downloaded and saved to marvel_comics_posters\Iron_Man_%282008_film%29_poster.jpg
Image downloaded and saved to marvel_comics_posters\Iron_Man_2_poster.jpg
Image downloaded and saved to marvel_comics_posters\Iron_Man_3_poster.jpg
Image downloaded and saved to marvel_comics_posters\Letitia_Wright_as_Shuri_in_Black_Pan
ther_poster.jpeg
Failed to download the image. Status code: 403
Image downloaded and saved to marvel_comics_posters\Martin_Freeman_as_Everett_K._Ross_in
_Black_Panther_poster.jpeg
Failed to download the image. Status code: 403
Image downloaded and saved to marvel_comics_posters\Shang-Chi_and_the_Legend_of_the_Ten_
Rings_poster.jpeg
Image downloaded and saved to marvel_comics_posters\Spider-Man_Far_From_Home_poster.jpg
Image downloaded and saved to marvel_comics_posters\Spider-Man_Homecoming_poster.jpg
Image downloaded and saved to marvel_comics_posters\Spider-Man_No_Way_Home_%E2%80%93_The
_More_Fun_Stuff_Version_poster.jpeg
Image downloaded and saved to marvel_comics_posters\Spider-Man_No_Way_Home_poster.jpg
```

```
Image downloaded and saved to marvel_comics_posters\The_Marvels_poster.jpg
Image downloaded and saved to marvel_comics_posters\Thor_%28film%29_poster.jpg
Image downloaded and saved to marvel_comics_posters\Thor_Love_and_Thunder_poster.jpeg
Image downloaded and saved to marvel_comics_posters\Thor_Ragnarok_poster.jpg
Image downloaded and saved to marvel_comics_posters\Thor_The_Dark_World_poster.jpg
Marvel Film posters downloaded.
```

In [ ]:
```python
# Data Collection - Music Album Covers
album_posters = "https://en.wikipedia.org/wiki/Category:Album_covers"
album_out_folder = "album_posters"
os.makedirs(album_out_folder, exist_ok=True)

response = requests.get(album_posters)

def download_album(url, save):
    try:
        response = requests.get(url)
        if(response.status_code) == 200:
            with open(path, 'wb') as file:
                file.write(response.content)
            print(f'Image Downloaded and Saved to {save}')
        else:
            print("Images failed to download")

    except Exception as e:
        print("Error Occured!")

if response.status_code == 200:
    soup = BeautifulSoup(response.text, 'html.parser')

    href_elements = soup.find_all('a')

    for href in href_elements:
        img_url = href.get('href')
        if img_url and img_url.startswith("/wiki/File:"):
            img_page_url = "https://en.wikipedia.org" + img_url
            img_page_response = requests.get(img_page_url)
            img_page_soup = BeautifulSoup(img_page_response.text, 'html.parser')
            img_element = img_page_soup.find('div', {'class': 'fullImageLink'})
            if img_element:
                img_link = img_element.find('a')['href']
                image_url = "https:" + img_link
                save_path = os.path.join(album_out_folder, os.path.basename(image_url))
                download_image(image_url, save_path)

print("Album Posters Downloaded")
```

In [ ]:
```python
# Data Collection - Book Covers
book_posters = "https://commons.wikimedia.org/wiki/Category:Book_covers"
book_out_folder = "book_posters"
os.makedirs(book_out_folder, exist_ok=True)

response = requests.get(book_posters)

def download_book(url, save):
    try:
        response = requests.get(url)
        if(response.status_code) == 200:
            with open(path, 'wb') as file:
                file.write(response.content)
            print(f'Image Downloaded and Saved to {save}')
        else:
            print("Images failed to download")

    except Exception as e:
```

```
        print("Error Occured!")

if response.status_code == 200:
    soup = BeautifulSoup(response.text, 'html.parser')

    href_elements = soup.find_all('a')

    for href in href_elements:
        img_url = href.get('href')
        if img_url and img_url.startswith("/wiki/File:"):
            img_page_url = "https://en.wikipedia.org" + img_url
            img_page_response = requests.get(img_page_url)
            img_page_soup = BeautifulSoup(img_page_response.text, 'html.parser')
            img_element = img_page_soup.find('div', {'class': 'fullImageLink'})
            if img_element:
                img_link = img_element.find('a')['href']
                image_url = "https:" + img_link
                save_path = os.path.join(book_out_folder, os.path.basename(image_url))
                download_image(image_url, save_path)

print("Book Posters Downloaded")
```

In [2]:
```
# Reading Book Covers Dataset
book_covers = pd.read_csv("./book_covers/main_dataset.csv")
book_covers.head()
```

Out[2]:

	image	name	author	format	book_depository_stars	price	cu
0	https://d1w7fb2mkkr3kw.cloudfront.net/assets/i...	This is Going to Hurt	Adam Kay	Paperback	4.5	7.6	
1	https://d1w7fb2mkkr3kw.cloudfront.net/assets/i...	Thinking, Fast and Slow	Daniel Kahneman	Paperback	4.0	11.5	
2	https://d1w7fb2mkkr3kw.cloudfront.net/assets/i...	When Breath Becomes Air	Paul Kalanithi	Paperback	4.5	9.05	
3	https://d1w7fb2mkkr3kw.cloudfront.net/assets/i...	The Happiness Trap	Russ Harris	Paperback	4.0	8.34	
4	https://d1w7fb2mkkr3kw.cloudfront.net/assets/i...	Man's Search For Meaning	Viktor E. Frankl	Paperback	4.5	9.66	

In [3]:
```
# Reading MCU Films Posters Dataset
movie_posters = pd.read_csv("./marvel_comics_posters/mcu_film_posters.csv")
display(movie_posters)
```

	image	label	path
0	1	Fantasy	./marvel_comics_posters/1.jpeg
1	2	Action	./marvel_comics_posters/2.jpeg
2	3	Action	./marvel_comics_posters/3.jpg
3	4	Adventure	./marvel_comics_posters/4.jpg
4	5	Action	./marvel_comics_posters/5.jpeg

5	6	Adventure	./marvel_comics_posters/6.jpg
6	7	Fantasy	./marvel_comics_posters/7.jpeg
7	8	Action	./marvel_comics_posters/8.jpg
8	9	Drama	./marvel_comics_posters/9.jpeg
9	10	Drama	./marvel_comics_posters/10.jpg
10	11	Drama	./marvel_comics_posters/11.jpg
11	12	Action	./marvel_comics_posters/12.jpg
12	13	Action	./marvel_comics_posters/13.jpg
13	14	Drama	./marvel_comics_posters/14.jpg
14	15	Action	./marvel_comics_posters/15.jpg
15	16	Drama	./marvel_comics_posters/16.jpg
16	17	Drama	./marvel_comics_posters/17.jpg
17	18	Action	./marvel_comics_posters/18.jpg
18	19	Fantasy	./marvel_comics_posters/19.jpg
19	20	SciFi	./marvel_comics_posters/20.jpg
20	21	Fantasy	./marvel_comics_posters/21.jpg
21	22	Fantasy	./marvel_comics_posters/22.jpg
22	23	Fantasy	./marvel_comics_posters/23.jpg
23	24	Fantasy	./marvel_comics_posters/24.jpg
24	25	Fantasy	./marvel_comics_posters/25.jpg
25	26	Action	./marvel_comics_posters/26.jpg
26	27	Action	./marvel_comics_posters/27.jpg
27	28	Action	./marvel_comics_posters/28.jpg
28	29	Action	./marvel_comics_posters/29.jpg
29	30	Action	./marvel_comics_posters/30.jpg
30	31	Fantasy	./marvel_comics_posters/31.jpg
31	32	Action	./marvel_comics_posters/32.jpg
32	33	Action	./marvel_comics_posters/33.jpg
33	34	Fantasy	./marvel_comics_posters/34.jpg
34	35	Adventure	./marvel_comics_posters/35.jpg
35	36	Adventure	./marvel_comics_posters/36.jpg
36	37	Fantasy	./marvel_comics_posters/37.jpg

In [4]:
```
# Reading Album Posters Dataset
music_covers = pd.read_csv("./album_posters/album_posters.csv")
display(music_covers)
```

	image	label	path
0	1	Rock	./album_posters/1.jpg

1	2	Rock	./album_posters/2.jpg
2	3	Electronic	./album_posters/3.jpg
3	4	Country	./album_posters/4.jpg
4	5	Jazz	./album_posters/5.jpg
5	6	Rock	./album_posters/6.jpg
6	7	Jazz	./album_posters/7.png
7	8	Jazz	./album_posters/8.jpg
8	9	Jazz	./album_posters/9.png
9	10	Jazz	./album_posters/10.jpg
10	11	Jazz	./album_posters/11.jpg
11	12	Electronic	./album_posters/12.jpg
12	13	Electronic	./album_posters/13.jpg
13	14	Jazz	./album_posters/14.jpeg
14	15	Rock	./album_posters/15.jpeg
15	16	Rock	./album_posters/16.jpg
16	17	Rock	./album_posters/17.jpg
17	18	Jazz	./album_posters/18.jpg
18	19	Electronic	./album_posters/19.jpg
19	20	Jazz	./album_posters/20.jpg
20	21	Jazz	./album_posters/21.jpg
21	22	Rock	./album_posters/22.jpg
22	23	Rock	./album_posters/23.jpg
23	24	Country	./album_posters/24.jpg
24	25	Country	./album_posters/25.jpg
25	26	Rock	./album_posters/26.png
26	27	Rock	./album_posters/27.png
27	28	Jazz	./album_posters/28.png
28	29	Country	./album_posters/29.jpg
29	30	Electronic	./album_posters/30.jpg
30	31	Electronic	./album_posters/31.png
31	32	Jazz	./album_posters/32.jpg

Data Quality Test

In [5]:
```
# Checking Null Values in Book Posters
check_na_book = book_covers.isnull().sum()
print(check_na_book)
```

```
image                      0
name                       0
author                   198
```

```
format                      33
book_depository_stars        0
price                        0
currency                     0
old_price                 5114
isbn                         0
category                     0
img_paths                    0
dtype: int64
```

In [6]:
```
check_na_music = music_covers.isnull().sum()
print(check_na_music)
```

```
image    0
label    0
path     0
dtype: int64
```

In [7]:
```
check_na_films = movie_posters.isnull().sum()
print(check_na_films)
```

```
image    0
label    0
path     0
dtype: int64
```

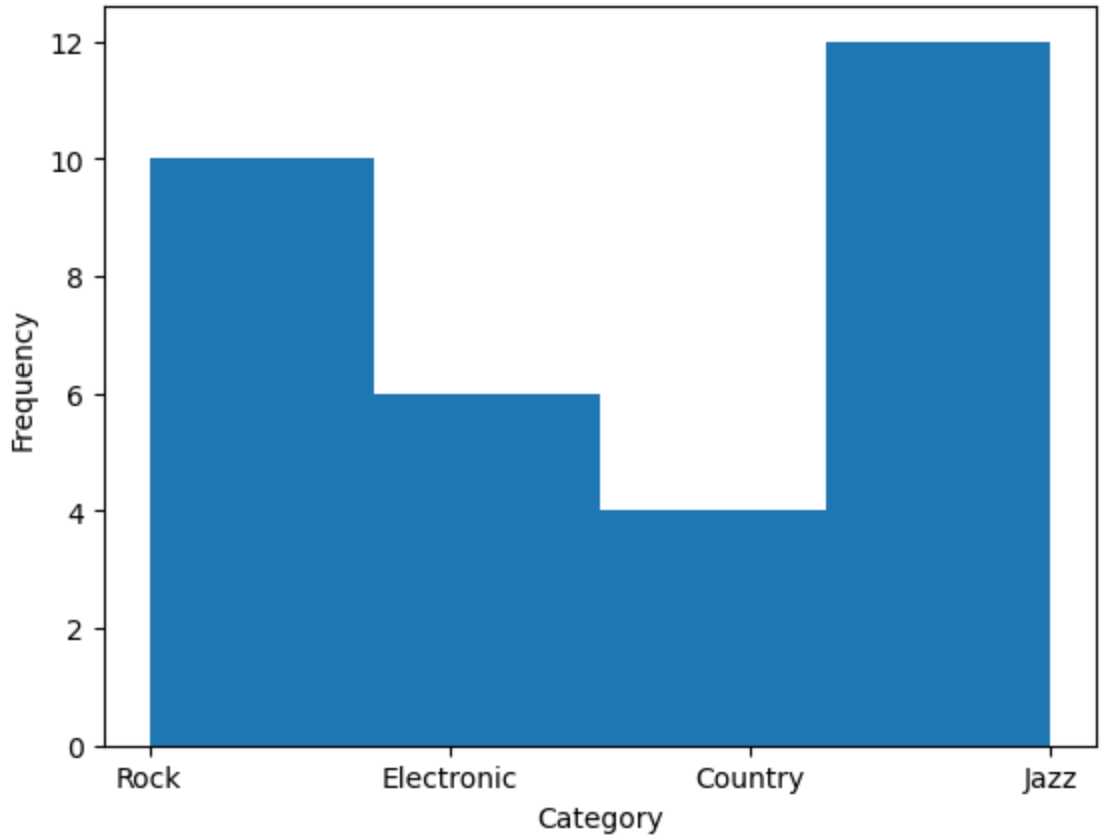
In [8]:
```
# Missing Value Imputation using "vz.median()"
fix_na_book = book_covers.fillna(vz.median(book_covers['old_price']))
check_fix = fix_na_book.isnull().sum()
print(check_fix)
```

```
image                    0
name                     0
author                   0
format                   0
book_depository_stars     0
price                    0
currency                 0
old_price                0
isbn                     0
category                 0
img_paths                0
dtype: int64
```

Data Distribution Analysis

In [9]:
```
# Albums Data Distribution Analysis
plt.hist(music_covers['label'],bins=4)
plt.xlabel("Category")
plt.ylabel("Frequency")

# Skew / Kurtosis Check
music_cat_count = music_covers['label'].value_counts()
check_music_kurt = vz.kurtosis(music_cat_count)
print("Kurtosis Value for Music Covers: ", check_music_kurt)
```

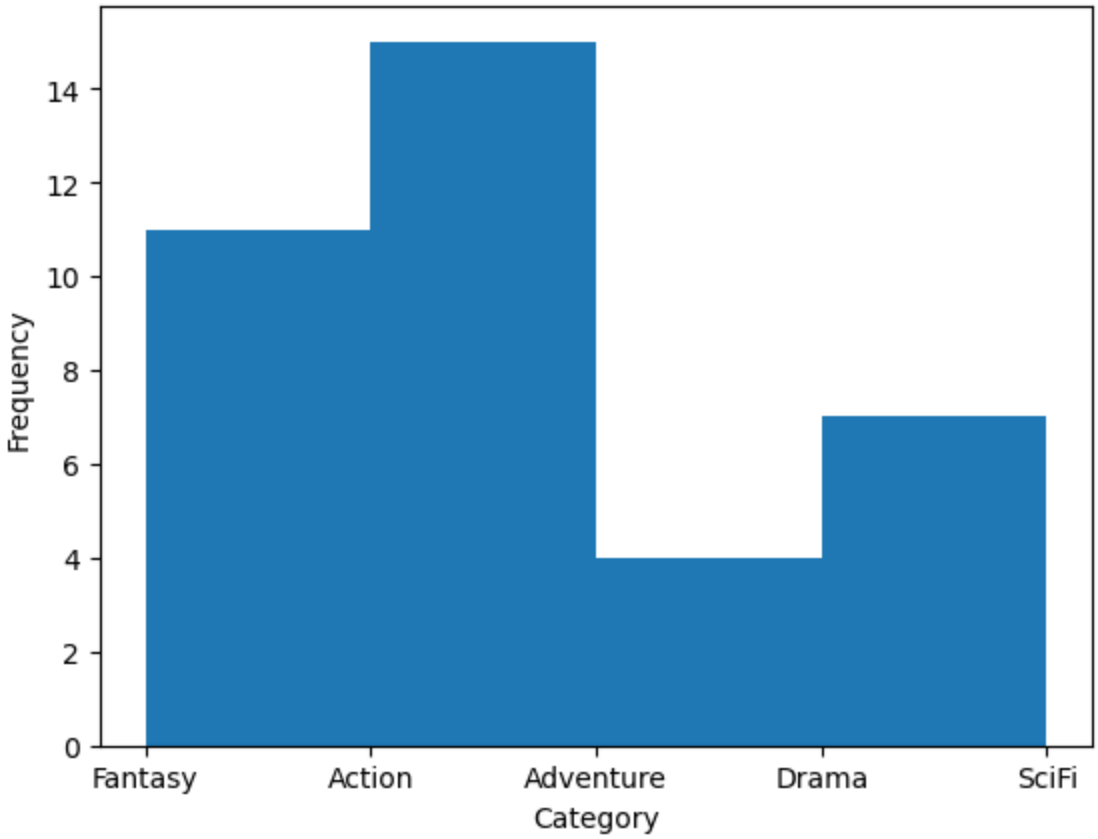
```
Kurtosis Value for Music Covers:  -1.9073486328125e-06
```



In [10]:

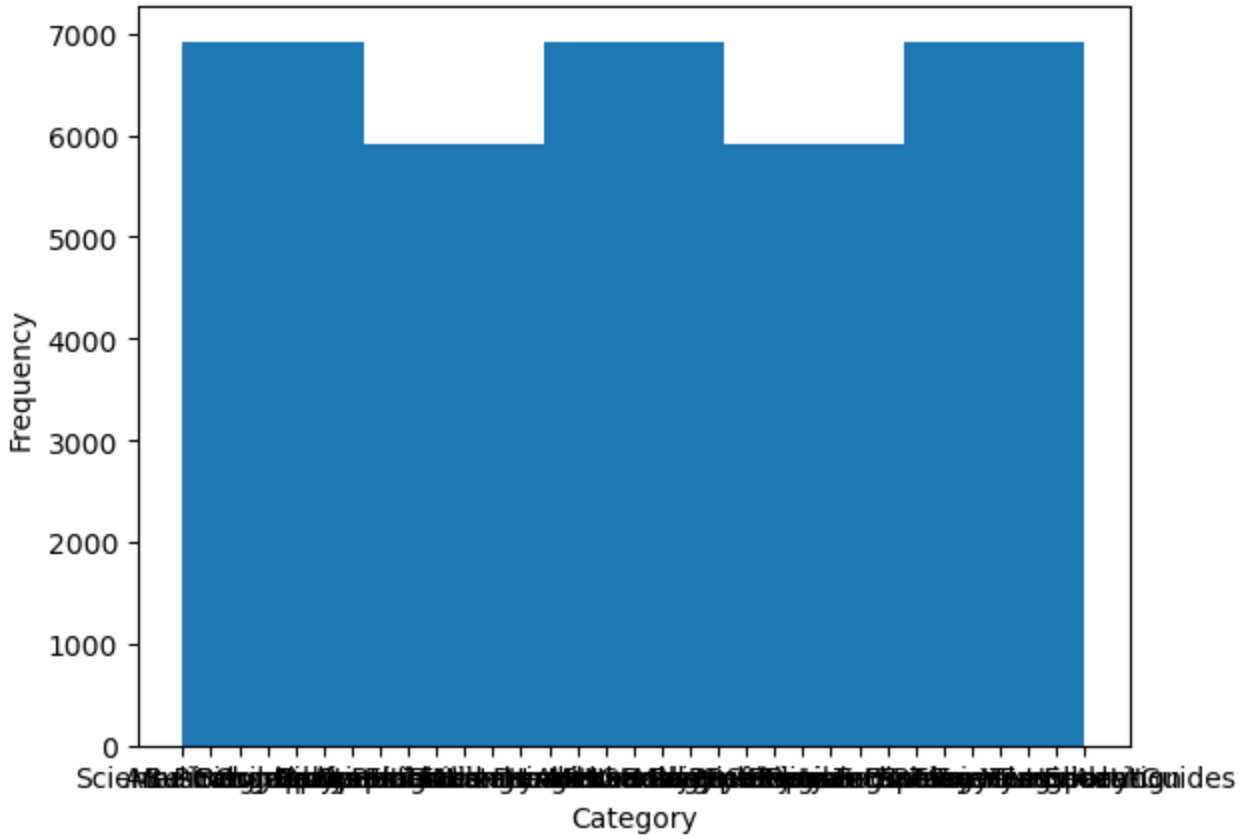
```
# Movies Data Distribution Analysis

plt.hist(movie_posters['label'],bins=4)
plt.xlabel("Category")
plt.ylabel("Frequency")

# Skew / Kurtosis Check
movie_cat_count = movie_posters['label'].value_counts()
check_movie_kurt = vz.kurtosis(movie_cat_count)
print("Kurtosis Value for Movie Posters: ", check_movie_kurt)
```

```
Kurtosis Value for Movie Posters:  -1.396358758211136e-06
```



In [11]:
```
# Book Covers Data Distribution Analysis
plt.hist(book_covers['category'],bins=5)
plt.xlabel("Category")
plt.ylabel("Frequency")

# Skew / Kurtosis Check
book_cat_count = book_covers['category'].value_counts()
check_book_kurt = vz.kurtosis(book_cat_count)
print("Kurtosis Value for Book Covers: ", check_book_kurt)
```

```
Kurtosis Value for Book Covers:  -442.4579617429399
```



1.2 Implement algorithm to extract features from image(-s) using k-means (k represents the number of color vectors in rgb values). In the feature vector, the color vectors are sorted from the ones representing the largest cluster to the smallest.

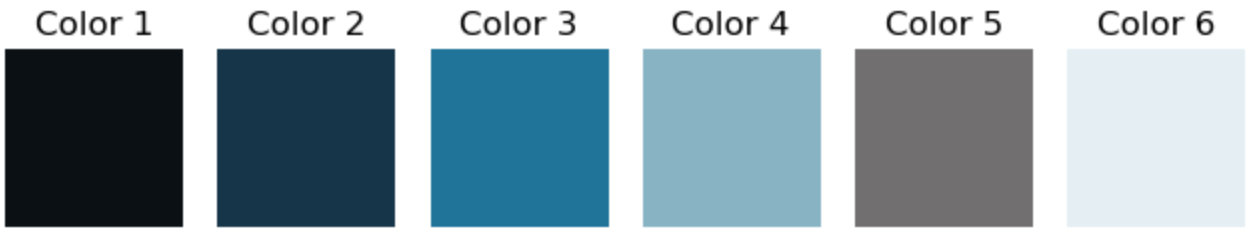
In [19]:
```
# Image Read - Movies
for index, images in movie_posters.iterrows():
    image = images['path']
    show_img = Image.open(image)
    display(show_img)
```

In [20]:
```
# Image Read - Music Covers
for index, images in music_covers.iterrows():
    image = images['path']
    show_img = Image.open(image)
    display(show_img)
```

In [14]:
```
# Implementing K-Means to extract features from images and updating the features vectors
get_movie_poster = Image.open("./book_covers/book-covers/Computing/0000001.jpg")

# Created an empty list for appending the cluster labels later
features = []

#Transforming image to numbers and wrap it inside an mutil-dimensional array
trans_to_array = np.array(get_movie_poster)
pixels = trans_to_array.reshape(-1,3)

# N-Clusters = 6
set_k = 6
k_means = KMeans(n_clusters = set_k, random_state=0).fit(pixels)

# Getting Cluster Labels
clust_labl = k_means.labels_
label_count = dict(Counter(clust_labl))
```

```
# Sorting Cluster Labels and Colors with Higher to Lower Count
sort_label = sorted(label_count, key=lambda k: label_count[k],reverse=True)
sort_colors = k_means.cluster_centers_[sort_label].astype(int)

# Flattening the Sorted Colors
feature_vec = sort_colors.flatten()
features.append(sort_colors.flatten())
get_movie_poster.show()

# Visualize the sorted color vectors as color swatches
plt.figure(figsize=(8, 6))
for i, color_vector in enumerate(sort_colors):
    plt.subplot(1, set_k, i + 1)
    color_patch = np.zeros((100, 100, 3), dtype=np.uint8)
    color_patch[:, :] = color_vector
    plt.imshow(color_patch)
    plt.axis('off')
    plt.title(f'Color {i + 1}')

print("Sorted Color Vectors: ", sort_colors)
print("Feature Vectors: ", feature_vec)

plt.show()
```

```
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
Sorted Color Vectors:  [[ 10  16  20]
 [ 23  53  72]
 [ 33 116 153]
 [135 179 195]
 [113 111 112]
 [229 239 243]]
Feature Vectors:  [ 10  16  20  23  53  72  33 116 153 135 179 195 113 111 112 229 239 2
43]
```



In [15]:

```
# Step 2: Extract features from images using K-Means
# Define the number of clusters for K-Means
k_means_clusters = 8

# Extract features from movie posters using K-Means
def extract_features(image_paths, k):
    features = []
    for path in image_paths:
        image = cv2.imread(path)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        h, w, _ = image.shape
        image = image.reshape(h * w, 3)

        kmeans = KMeans(n_clusters=k)
        kmeans.fit(image)
        labels = kmeans.labels_

        # Finding the Centroid in each clusters
        cluster_centers = kmeans.cluster_centers_
```

```python
        # Sorted cluster centers by the size of the cluster
        cluster_sizes = np.bincount(labels)
        sorted_cluster_indices = np.argsort(cluster_sizes)[::-1]
        sorted_cluster_centers = cluster_centers[sorted_cluster_indices]

        features.append(sorted_cluster_centers.flatten())

    return np.array(features)

image_paths = movie_posters['path'].tolist()
features = extract_features(image_paths, k_means_clusters)
```

```
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
```

```
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
```

```
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

1.3 Use KNN algorithm to evaluate the extracted features using the main category as label. Based on the results, select the appropriate number of clusters for feature extraction.

In [16]:

```python
image_paths = ["./marvel_comics_posters/1.jpeg", "./marvel_comics_posters/2.jpeg", "./ma
labels = ["Fantasy", "Action", "Action"]

# Implement algorithm to extract features from images using k-means
def extract_features(image_paths, k):
    features = []
    for path in image_paths:
        img = Image.open(path)
        pixels = np.array(img).reshape(-1, 3)

        kmeans = KMeans(n_clusters=k)
        kmeans.fit(pixels)
        labels = kmeans.labels_
        cluster_centers = kmeans.cluster_centers_

        # Sort cluster_centers by the size of the cluster
        cluster_sizes = np.bincount(labels)
        sorted_cluster_indices = np.argsort(cluster_sizes)[::-1]
        sorted_cluster_centers = cluster_centers[sorted_cluster_indices]

        features.append(sorted_cluster_centers.flatten())

    return np.array(features)

# Using KNN algorithm to evaluate the extracted features using the main category as a la
def knn_evaluate(features, labels, k_neighbors):
    knn = KNeighborsClassifier(n_neighbors=k_neighbors)
    knn.fit(features, labels)
    accuracy = knn.score(features, labels)
    return accuracy
```

```
# Select an appropriate number of clusters for feature extraction
k = 5  # Number of clusters for K-Means
k_neighbors = 3  # Number of neighbors for KNN

# Extract features from images
features = extract_features(image_paths, k)

# Evaluate with KNN
accuracy = knn_evaluate(features, labels, k_neighbors)
print(f"KNN Accuracy: {accuracy * 100:.2f}%")
```

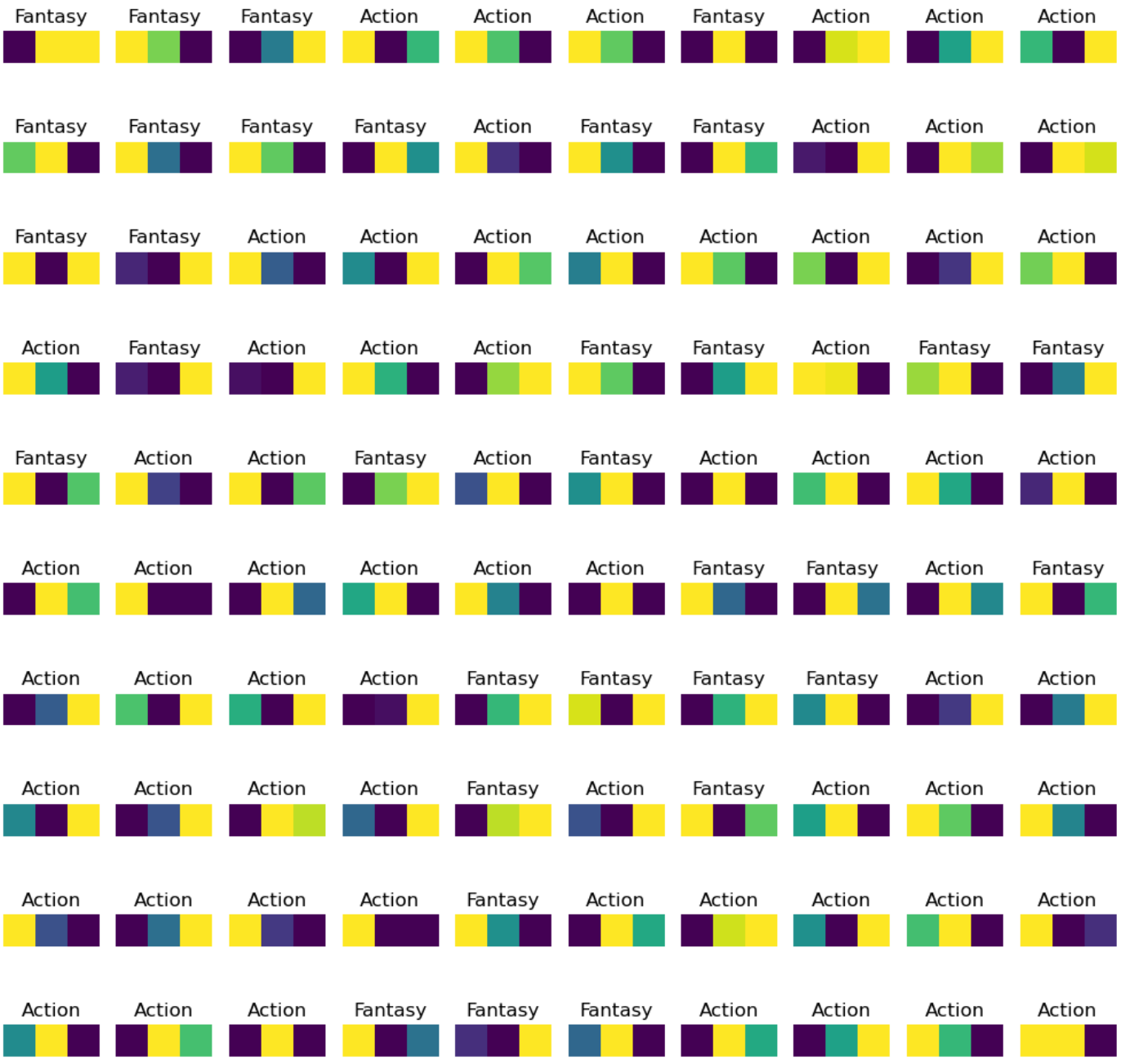
```
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ravin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
KNN Accuracy: 66.67%
```

1.4 Implement SOM for the dataset using extracted features as an input vector. Analyze the results (provide grids with colors representing the most frequent category of the node, the dominant color for the node, ...). Comment on the results whether the pattern between categories is logical, the similarity makes sense, etc.

In [17]:

```
# SOM grid dimensions
grid_width = 10
grid_height = 10
input_size = features.shape[1]

# Creating a SOM instance
som = MiniSom(grid_width, grid_height, input_size, sigma=1.0, learning_rate=0.5)

# Initialize the weights
som.random_weights_init(features)

# Training SOM
num_epochs = 1000
som.train_batch(features, num_epochs)

# Creating a grid of nodes and their associated categories
node_categories = np.empty((grid_width, grid_height), dtype=object)
node_colors = np.empty((grid_width, grid_height, 3), dtype=int)

# Determining the most frequent category and dominant color for each node
for i in range(grid_width):
    for j in range(grid_height):
        node_feature_weights = som.get_weights()[i, j]
        distances = np.linalg.norm(node_feature_weights - features, axis=1)

        if len(distances) > 0:
            closest_sample_index = np.argmin(distances)
            node_categories[i, j] = labels[closest_sample_index]

            # Find the most common color in the cluster
            cluster_indices = np.where(labels == node_categories[i, j])
```

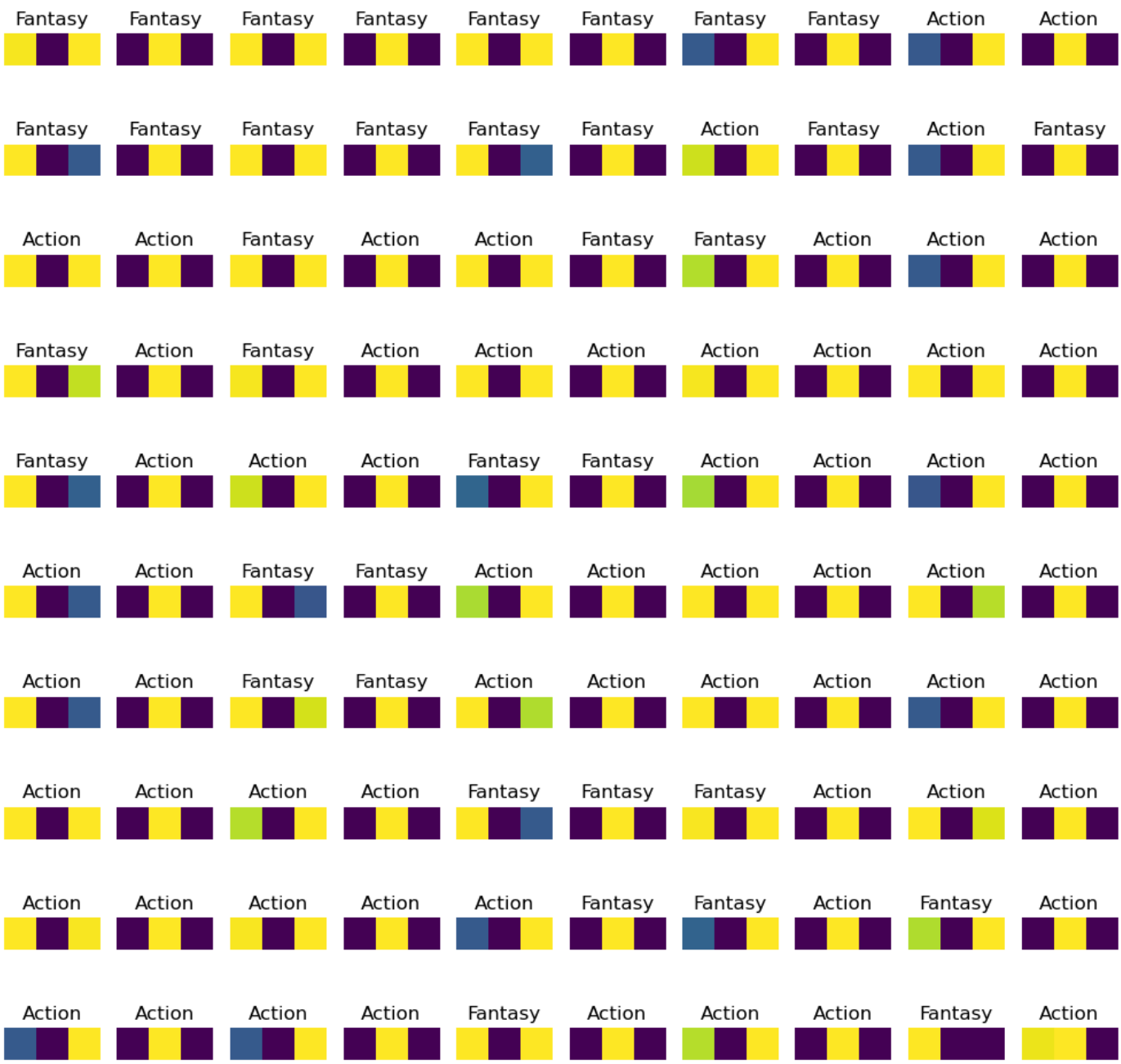
```
            cluster_colors = features[cluster_indices]

            if len(cluster_colors) > 0:
                # Calculate the mode (most common color) for the cluster
                most_common_color, _ = vz.mode(cluster_colors)
                most_common_color = most_common_color.flatten().astype(int)
                node_colors[i, j] = most_common_color

# Visualize the SOM grid
plt.figure(figsize=(10, 10))
for i in range(grid_width):
    for j in range(grid_height):
        plt.subplot(grid_width, grid_height, i * grid_width + j + 1)
        plt.imshow([node_colors[i, j]])
        plt.title(node_categories[i, j])
        plt.axis('off')

plt.tight_layout()
plt.show()
```



In [18]:

```
grid_width = 10
grid_height = 10
input_size = features.shape[1]  # Size of feature vectors
```

```
som = MiniSom(grid_width, grid_height, input_size, sigma=1.0, learning_rate=0.5)

som.random_weights_init(features)

num_epochs = 1000  # You may adjust this based on your data
som.train_batch(features, num_epochs)

node_categories = np.empty((grid_width, grid_height), dtype=object)
node_colors = np.empty((grid_width, grid_height, 3), dtype=int)

for i in range(grid_width):
    for j in range(grid_height):
        node_feature_weights = som.get_weights()[i, j]
        distances = np.linalg.norm(node_feature_weights - features, axis=1)

        if len(distances) > 0:
            closest_sample_index = np.argmin(distances)
            node_categories[i, j] = labels[closest_sample_index]


            cluster_indices = np.where(labels == node_categories[i, j])
            cluster_colors = features[cluster_indices]

            if len(cluster_colors) > 0:
                most_common_color, _ = mode(cluster_colors,axis=0)
                most_common_color = most_common_color.flatten().astype(int)
                node_colors[i, j] = most_common_color

plt.figure(figsize=(10, 10))
for i in range(grid_width):
    for j in range(grid_height):
        plt.subplot(grid_width, grid_height, i * grid_width + j + 1)
        plt.imshow([node_colors[i, j]])
        plt.title(node_categories[i, j])
        plt.axis('off')

plt.tight_layout()
plt.show()
```


Fantasy Fantasy Fantasy Fantasy Fantasy Fantasy Fantasy Fantasy Action Action
Fantasy Fantasy Fantasy Fantasy Fantasy Fantasy Action Fantasy Action Fantasy
Action Action Fantasy Action Action Fantasy Fantasy Action Action Action
Fantasy Action Fantasy Action Action Action Action Action Action Action
Fantasy Action Action Action Fantasy Fantasy Action Action Action Action
Action Action Fantasy Fantasy Action Action Action Action Action Action
Action Action Fantasy Fantasy Action Action Action Action Action Action
Action Action Action Action Fantasy Fantasy Fantasy Action Action Action
Action Action Action Action Action Fantasy Fantasy Action Fantasy Action
Action Action Action Action Fantasy Action Action Action Fantasy Action