23/24 Introduction to Data Science
In-Class Assignment Portfolio - Exercise 4: Explore the use of API

0 Introduction

**Related Course Material: Week 2 Acquiring and Clearing Data, Week 8 Unsupervised Learning*

In Week 2, we explored sourcing data from public APIs using Python's requests package. This exercise further looked into several different APIs to find out how useful and efficient they are in retrieving information and creating datasets.

The data (both text and image) retrieved were used in building a simple web program and to explore image clustering tasks based on the content covered in Week 8 on Unsupervised Learning.

The following APIs were used in this exercise:

1) The Nobel Prize API (https://www.nobelprize.org/about/developer-zone-2/) – free to use according to the CC0 license
2) OpenWeather API (https://openweathermap.org/api) – free API calls with subscription
3) The Cat API (https://thecatapi.com/) – free access
4) Unsplash API (https://unsplash.com/developers) - free API calls with subscription

2 Getting Data from APIs

I have made attempts to retrieve different kinds of data here. The Nobel Prize API and the OpenWeather API were to retrieve text information about Nobel Prizes and weather, and the Cat API and Unsplash API were used to retrieve image data.

[Text Data]

- The Nobel Prize API – Build Nobel Prizes (1901-2023) Dataset

    - Endpoint 1: https://api.nobelprize.org/2.1/laureates; returns all information about Laureates and Nobel Prizes.
    - Endpoint 2: https://api.nobelprize.org/2.1/nobelPrizes; returns a shorter result and the Laureates endpoint needs to be called as well to get the full response.

    Started with retrieving some specific data. I'm interested in finding out who won the Chemistry Nobel Prize last year. By specifying category = 'che' and year = '2023' using endpoint 2, I was able to retrieve the information I wanted and directly print the desired results with some simple formatting and selection: *'There are 3 Nobel Prize winners in Chemistry in 2023: ['Moungi G. Bawendi', 'Louis E. Brus', 'Aleksey Yekimov'*].

```
endpoint = "http://api.nobelprize.org/2.0/nobelPrize/"
category = "che" # che, eco, lit, pea, phy, med
year = "2023" # choose from 1901 to 2023

url = endpoint + category + "/" + year
response = requests.get(url)
response.status_code
```

✓ 0.5s Python

```
200
```

```
response.text
```

✓ 0.0s Python

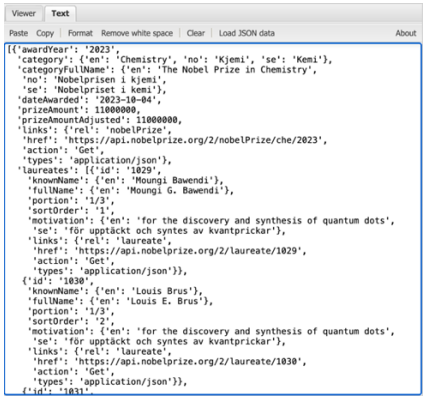
```
'[{"awardYear":"2023","category":{"en":"Chemistry","no":"Kjemi","se":"Kemi"},"categoryFullName":{"en":"The Nobel
```

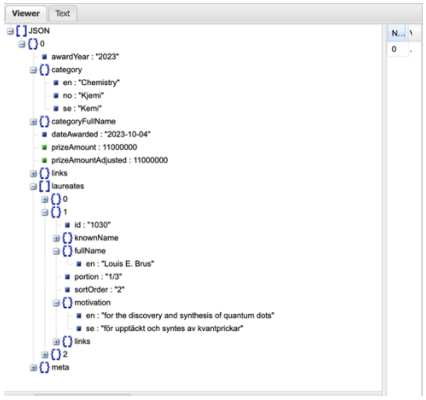
```
results = response.json()
name_list = [laureate['fullName']['en'] for laureate in results[0]['laureates']]
print(f"There are {len(results[0]['laureates'])} nobel prizes winner in {results[0]['awardYear']} in {results
```

✓ 0.0s Python

```
There are 3 nobel prizes winner in 2023 in Chemistry: ['Moungi G. Bawendi', 'Louis E. Brus', 'Aleksey Yekimov ']
```

Note it can be hard to read the JSON text especially when there are lot of information – therefore I found an online JSON Viewer tool (https://jsonviewer.stack.hu/) that can transform JSON text into an easily readable format, which can be helpful when setting rules for formatting and selecting target information.





In this case, it's not that efficient to use the API if I only want some specific or few pieces of data (google can be much faster), it would be, however, much more useful if I want to retrieve a large amount of information or build a dataset, e.g., build a dataset contains information about all Nobel Prize winners from 1901 to 2023.

Without specifying anything, endpoint 2 can be used directly to retrieve all Nobel Prize information. However, the default setting has a limit of 25 entries, so I explicitly added a limit of 1000 to ensure I could retrieve all the data.

```
# default endpoint setting only returns 25 entries - add a limit of 1000 to show all resutls
prizes_url = "http://api.nobelprize.org/2.0/nobelPrizes?limit=1000"
prizes_response = requests.get(prizes_url)
prizes_response.status_code
```

✓ 0.1s

```
200
```

Using *pd.json_normalize()* allows me to put the retrieved data into a DataFrame, and I have only kept those columns related to award year, date, prize, and category. The converted df has 670 rows, and some rows have missing data. To investigate, I filtered out those rows and iterated through each row to print the relevant motivation information. It turns out that those rows with missing laureates are the years with no Nobel Prize awarded. Since I am only interested in having a dataset of prize winners, those rows can be safely removed.

```
No Nobel Prize was awarded this year. The prize money was allocated to the Special Fund of this prize section.
No Nobel Prize was awarded this year. The prize money was allocated to the Special Fund of this prize section.
No Nobel Prize was awarded this year. The prize money was allocated to the Special Fund of this prize section.
No Nobel Prize was awarded this year. 1/3 of the prize money was allocated to the main fund and 2/3 was allocated to the special fund of this prize section.
No Nobel Prize was awarded this year. 1/3 of the prize money was allocated to the main fund and 2/3 was allocated to the special fund of this prize section.
No Nobel Prize was awarded this year. 1/3 of the prize money was allocated to the main fund and 2/3 was allocated to the special fund of this prize section.
...
No Nobel Prize was awarded this year. 1/3 of the prize money was allocated to the main fund and 2/3 was allocated to the special fund of this prize section.
No Nobel Prize was awarded this year. The prize money was allocated to the Special Fund of this prize section.
No Nobel Prize was awarded this year. 1/3 of the prize money was allocated to the main fund and 2/3 was allocated to the special fund of this prize section.
No Nobel Prize was awarded this year. The prize money for 1972 was allocated to the Main Fund.
```

The new df has 621 rows, which matches the information posted on the nobelprize.org - *"between 1901 and 2023… were awarded 621 times to 1,000 people and organisations..."*, meaning that we have successfully extracted all award information between 1901 to 2023.

	awardYear	dateAwarded	laureates	prizeAmountAdjusted	category.en	links.href
0	1901	1901-11-12	[{'id': '160', 'knownName': {'en': 'Jacobus H....	9704878	Chemistry	https://api.nobelprize.org/2/nobelPrize/che/1901
1	1901	1901-11-14	[{'id': '569', 'knownName': {'en': 'Sully Prud...	9704878	Literature	https://api.nobelprize.org/2/nobelPrize/lit/1901
2	1901	1901-12-10	[{'id': '462', 'knownName': {'en': 'Henry Duna...	9704878	Peace	https://api.nobelprize.org/2/nobelPrize/pea/1901
3	1901	1901-11-12	[{'id': '1', 'knownName': {'en': 'Wilhelm Conr...	9704878	Physics	https://api.nobelprize.org/2/nobelPrize/phy/1901
4	1901	1901-10-30	[{'id': '293', 'knownName': {'en': 'Emil von B...	9704878	Physiology or Medicine	https://api.nobelprize.org/2/nobelPrize/med/1901
...	...	...	...	...	...	...
665	2023	2023-10-09	[{'id': '1034', 'knownName': {'en': 'Claudia G...	11000000	Economic Sciences	https://api.nobelprize.org/2/nobelPrize/eco/2023
666	2023	2023-10-05	[{'id': '1032', 'knownName': {'en': 'Jon Fosse...	11000000	Literature	https://api.nobelprize.org/2/nobelPrize/lit/2023
667	2023	2023-10-06	[{'id': '1033', 'knownName': {'en': 'Narges Mo...	11000000	Peace	https://api.nobelprize.org/2/nobelPrize/pea/2023
668	2023	2023-10-03	[{'id': '1026', 'knownName': {'en': 'Pierre Ag...	11000000	Physics	https://api.nobelprize.org/2/nobelPrize/phy/2023
669	2023	2023-10-02	[{'id': '1024', 'knownName': {'en': 'Katalin K...	11000000	Physiology or Medicine	https://api.nobelprize.org/2/nobelPrize/med/2023

621 rows × 6 columns

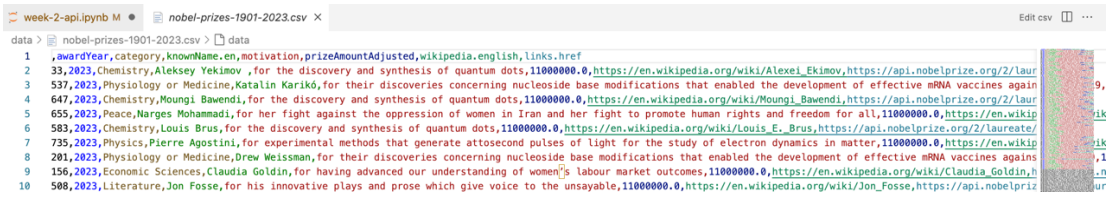
However, in this df, one entry could sometimes contain two or more laureates, which makes it a bit complicated to extract information and form a new df with each row corresponding to each laureate, therefore, I also tried another endpoint (endpoint 1) to see if it's easier to obtain a clean dataset. This time it returns 992 entries, which also matches the number posted on nobelprize.org - *"...with some receiving the Nobel Prize more than once, this makes a total of 965 individuals and 27 organisations..."*.

I have reformatted and reordered the df, as well as extracted some key information from the 'nobelPrizes' column and replaced it with new columns. The updated df is shown below, ordered by award year in descending order, followed by category, winner name, motivation, prize amount, relevant Wikipedia link, and API link.

	awardYear	category	knownName.en	motivation	prizeAmountAdjusted	wikipedia.english	links.href
33	2023	Chemistry	Aleksey Yekimov	for the discovery and synthesis of quantum dots	11000000.0	https://en.wikipedia.org/wiki/Alexei_Ekimov	https://api.nobelprize.org/2/laureate/1031
537	2023	Physiology or Medicine	Katalin Karikó	for their discoveries concerning nucleoside ba...	11000000.0	https://en.wikipedia.org/wiki/Katalin_Karikó	https://api.nobelprize.org/2/laureate/1024
647	2023	Chemistry	Moungi Bawendi	for the discovery and synthesis of quantum dots	11000000.0	https://en.wikipedia.org/wiki/Moungi_Bawendi	https://api.nobelprize.org/2/laureate/1029
655	2023	Peace	Narges Mohammadi	for her fight against the oppression of women ...	11000000.0	https://en.wikipedia.org/wiki/Narges_Mohammadi	https://api.nobelprize.org/2/laureate/1033
583	2023	Chemistry	Louis Brus	for the discovery and synthesis of quantum dots	11000000.0	https://en.wikipedia.org/wiki/Louis_E._Brus	https://api.nobelprize.org/2/laureate/1030
...	...	...	...	...	...	...	...
882	1901	Literature	Sully Prudhomme	in special recognition of his poetic compositi...	9704878.0	https://en.wikipedia.org/wiki/Sully_Prudhomme	https://api.nobelprize.org/2/laureate/569
283	1901	Peace	Frédéric Passy	for his lifelong work for international peace ...	9704878.0	https://en.wikipedia.org/wiki/Frédéric_Passy	https://api.nobelprize.org/2/laureate/463
441	1901	Chemistry	Jacobus H. van 't Hoff	in recognition of the extraordinary services h...	9704878.0	https://en.wikipedia.org/wiki/Jacobus_Henricus...	https://api.nobelprize.org/2/laureate/160
949	1901	Physics	Wilhelm Conrad Röntgen	in recognition of the extraordinary services h...	9704878.0	https://en.wikipedia.org/wiki/Wilhelm_Röntgen	https://api.nobelprize.org/2/laureate/1
232	1901	Physiology or Medicine	Emil von Behring	for his work on serum therapy, especially its ...	9704878.0	https://en.wikipedia.org/wiki/Emil_Adolf_von_B...	https://api.nobelprize.org/2/laureate/293

992 rows × 7 columns

The updated df has also been converted into a CSV file which can be used for other data analysis tasks. The file can be found via the provided GitHub link.



Overall, APIs can be very useful in retrieving large amounts of text data and creating datasets. Nevertheless, while this is very effective in terms of data collection, the data we retrieve can still be unformatted (e.g., one column contains multiple features) or contain errors and missing data. The data will also carry biases which can be even harder for us to spot as we did not go through the initial data collection. So, it is important to go through the original documentation of the API and even check the sources whenever possible, to make sure that we understand the data and can pre-process them accordingly before doing analysis.

- OpenWeather API – Getting Current Weather / Weather Forecast Data

OpenWeather API is very famous for providing weather data. It offers free API calls with a subscription. I have tried two of its free API calls – the current weather API call and the 5-day/3-hour forecast API call, and have obtained some weather information for London by specifying the latitude and longitude.

- The Current Weather API call returns current weather information including weather description, temperature, pressure, humidity, etc.
- The 5 day/3 hour forecast API call returns a 5-day forecast, providing weather forecast data with 3-hour steps (00:00, 03:00, 06:00, 09:00, 12:00, 15:00, 18:00, 21:00 of each day). Whenever the API is called, it returns 40 sets of data, and the first set of data is the time of the one closest to it above, e.g., if I call this API at 7:00 in the morning, the first set of data returned is the forecast data at 9:00AM.

Not too surprisingly, the current weather description in London is 'overcast clouds'.

```
# London, can be changed to other location using different lat&lon
lat = 51.507351
lon = -0.127758

# apikey obtained from openweather api
api_key = "93146e366d37fc681f5daf1f33371a50"

current_weather_url = f"https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={api_key}"
current_weather_response = requests.get(current_weather_url)
current_weather_response.status_code
```

✓ 0.0s Python

```
200
```

```
current_weather_results = current_weather_response.json()
current_weather_results
```

✓ 0.0s Python

```
{'coord': {'lon': -0.1276, 'lat': 51.5072},
 'weather': [{'id': 804,
   'main': 'Clouds',
   'description': 'overcast clouds',
   'icon': '04n'}],
```

and the nearest forecast weather description is 'broken clouds'…

```
lat = 51.507351
lon = -0.127758
count = 1 # to check the closet forecast
api_key = "93146e366d37fc681f5daf1f33371a50"

weather_forecast_url = f"https://api.openweathermap.org/data/2.5/forecast?lat={lat}&lon={lon}&cnt={count}&appid={api_key}"
weather_forecast_response = requests.get(weather_forecast_url)
weather_forecast_response.status_code
```

✓ 0.1s Python

```
200
```

```
weather_forecast_results = weather_forecast_response.json()
weather_forecast_results
```

✓ 0.0s Python


```
'message': 0,
'cnt': 1,
'list': [{'dt': 1709078400,
  'main': {'temp': 280.61,
   'feels_like': 279.3,
   'temp_min': 280.39,
   'temp_max': 280.61,
   'pressure': 1019,
   'sea_level': 1019,
   'grnd_level': 1017,
   'humidity': 82,
   'temp_kf': 0.22},
  'weather': [{'id': 803,
    'main': 'Clouds',
    'description': 'broken clouds',
```


[Image Data]

- The Cat API – Get a random cat image

  The Cat API returns random cat images. The process of calling the API and retrieving the image data is the same as obtaining text data, the url of the image will be retrieved in JSON format, which can be downloaded, opened in a browser, or directly displayed in JupyterNotebook using `IPython.display`.

```
cat_url = 'https://api.thecatapi.com/v1/images/search'
cat_response = requests.get(cat_url)
if cat_response.status_code == 200:
    # every time it returns a random cat img!
    cat_result = cat_response.json()
    cat_url = cat_result[0]['url']
    display(Image(url=cat_url, width=300))
else:
    print(cat_response.status_code)
```

✓ 0.1s

```
cat_url = 'https://api.thecatapi.com/v1/images/search'
cat_response = requests.get(cat_url)
if cat_response.status_code == 200:
    # every time it returns a random cat img!
    cat_result = cat_response.json()
    cat_url = cat_result[0]['url']
    display(Image(url=cat_url, width=300))
else:
    print(cat_response.status_code)
```

✓ 0.2s

- Unsplash – Build an Image Dataset

  Multiple images can also be retrieved all at once to create an image dataset. In week 8, we learned about image clustering and UMAP visualisation which need to be performed on an image dataset. Since the Cat API provides only cat images and lacks diversity, I have found the Unsplash API, which offers diverse high-quality photos.

  Similar to the OpenWeather API, Unsplash also requires an API key to make API calls. It is free to use but has a limitation of 50 requests per hour, fortunately, it is sufficient for me to obtain a small image dataset of around 300 images.

  The process of making the API call, however, needs to be run several times due to the Unsplash API's limitation of 30 images per request. Because the images are not obtained all at once, there might be some duplicated images retrieved already, I used

regex (learned last term in the NLP course) to identify the image id in each url (there is a unique id for each image) and saved only the unique images into a folder.

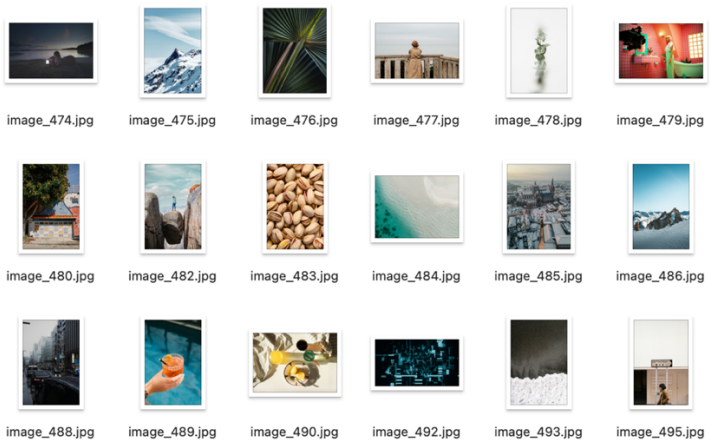
```python
for index, photo in enumerate(unplash_data, start=691): # start from 1, then change to 31, 61, 91, ...
    image_url = photo['urls']['regular']
    # regex tool: https://regex101.com/
    # search id between photo- and -, return the first match saved as the image id
    image_id = re.search(r'photo-([0-9a-f]+)-', image_url).group(1)

    # save the image if its id is not yet in the list
    if not any(image_id in url for url in images):
        images.append(image_url)
        # get binary data of the image
        # .content vs .text reference code: https://stackoverflow.com/questions/17011357/what-is-the-difference-between-content-and-text
        image_data = requests.get(image_url).content
        # download the image to local folder
        # reference code: https://blog.apify.com/save-image-python/, 'wb' for binary mode - save binary files such as image
        with open(f'images/unsplash_images/image_{index}.jpg', 'wb') as f:
            f.write(image_data)
            print(f"Image {index} downloaded")
    else:
        print(f"Image {index} with an ID of {image_id} already exists, skipping download")
```

✓ 1.0s

```
Image 691 downloaded
Image 692 with an ID of 1705449589011 already exists, skipping download
Image 693 downloaded
Image 694 with an ID of 1705669230343 already exists, skipping download
Image 695 with an ID of 1706403877657 already exists, skipping download
```

I have tried to retrieve 720 images in total and got 558 unique images at the end.



3 Building a Simple Webpage using APIs – 'your weather cat'

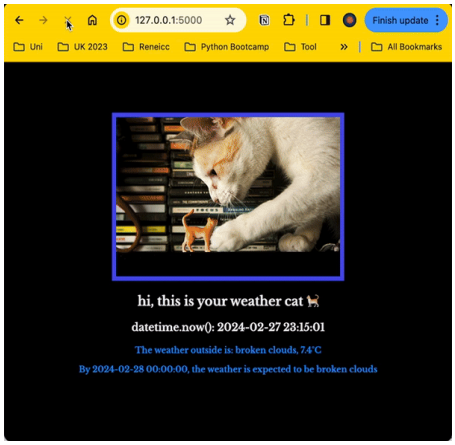
APIs allow accessing and interacting with various services and databases. With the combination of different APIs, there are lots of fun programs and applications we can make.

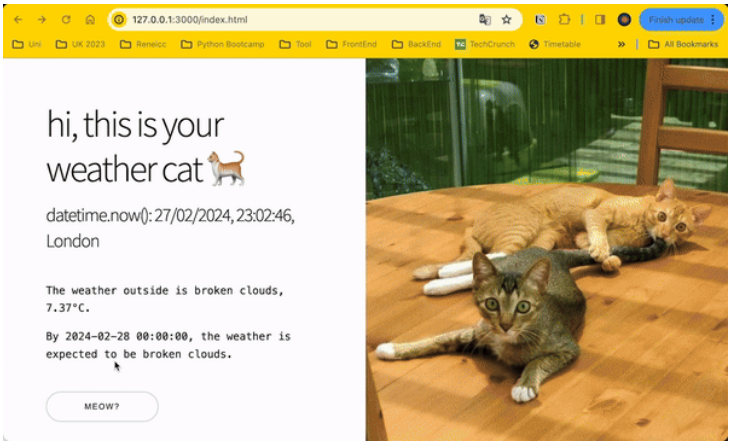
Here, I attempted to make a simple webpage that shows the current datetime, current weather, weather forecast, and random cat image every time the page is refreshed by integrating The Cat API and OpenWeather API into HTML/CSS code.

I have included two versions here:

1. The first version is a very simple HTML/CSS webpage (HTML/CSS learnt from Udemy Python Course - 100 Days of Code: The Complete Python Pro Bootcamp), focusing on utilising APIs and displaying information with little design effort.
2. The second version is based on a free web template I found on https://html5up.net/, which is free for use under the CCA 3.0 license. I integrated the APIs and relevant code into the template to create a visually enhanced version (the template came with original HTML/CSS/JavaScript, modification to the code was made with the help of ChatGPT - especially for transferring python code to javascript code)

The .gif of the two versions can be found below:

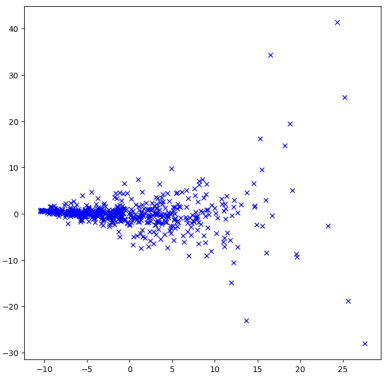




The relevant HTML/CSS/Python/JavaScript code can all be found in the submitted zipped file or on GitHub (https://git.arts.ac.uk/23001934/ds-portfolio / https://git.arts.ac.uk/23001934/ds-portfolio-weather-cat), along with gifs and videos.

4 Image Dataset for Image Clustering

The image dataset obtained using Unsplash API is saved in a folder and can be directly used for the image Clustering task.



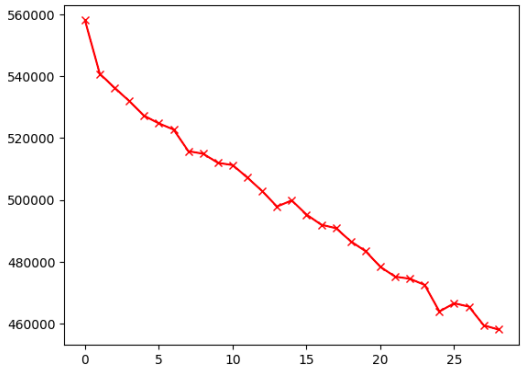


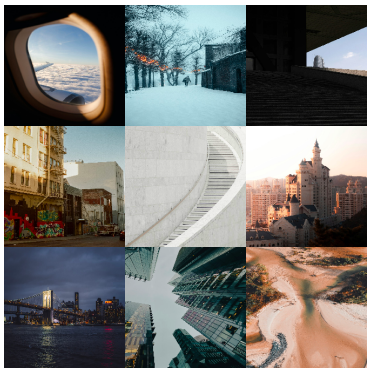
Image clustering is performed on this image dataset using MobileNetV2 for feature extraction and K-means for clustering. In terms of finding the optimal k, the elbow method was used but the result doesn't really show a clear elbow point, which is not uncommon for real-world datasets (Tomar, 2023) but at least allows me to have some good guesses (i.e. k=7, k=13, k=24).

I have tried different k numbers, at the end, k = 13 with num_dimensions_to_reduce_to = 20 gave some pretty meaningful results:

The picture on the right shows the images closest to the centre for each cluster.

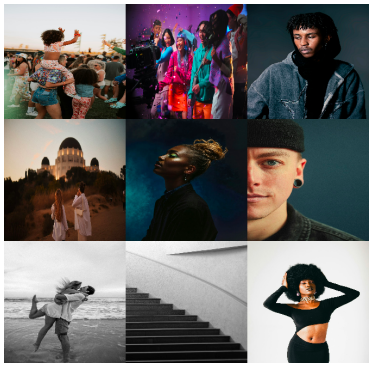
Cluster 0 seems to be related to buildings; Cluster 1 relates to plants; Images in clusters 2 and 4 mostly are human figures; Cluster 6 relates to food; Clusters 8 and 9 are about natural views, and Cluster 12 seems to have many abstract images.

**Detailed images can be found on GitHub – /ds-portfolio/images/image clustering results*



Images Nearest the Centre of Cluster 0

Images Nearest the Centre of Cluster 1

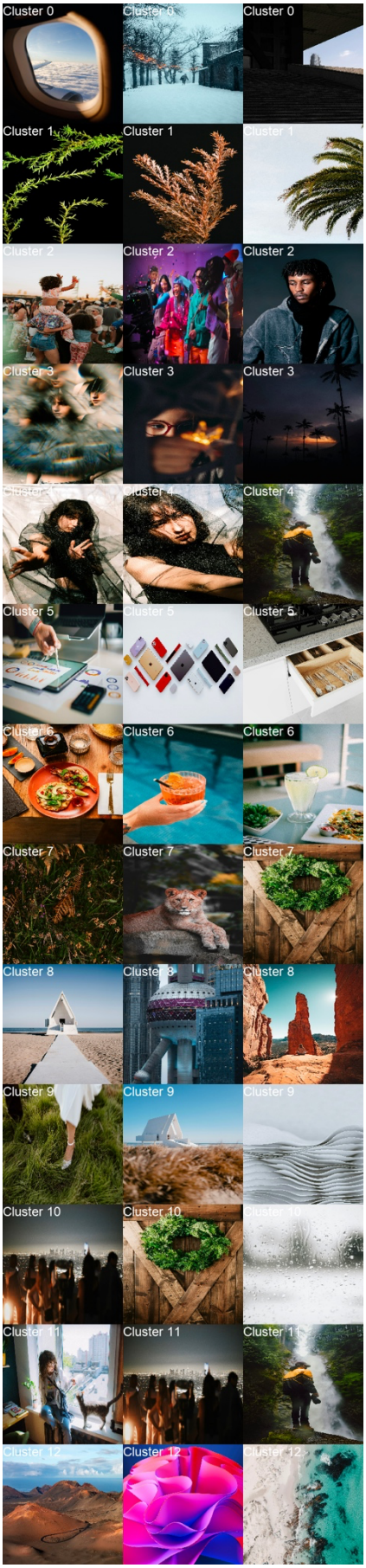


Images Nearest the Centre of Cluster 2

Images Nearest the Centre of Cluster 10

Images Nearest the Centre of Cluster 6

Images Nearest the Centre of Cluster 8



Nevertheless, the clustering is not perfect here, e.g., Cluster 3 with most human figures, also has an image of stairs; Cluster 8 with most pictures related to the natural view, has an image of city vie; Clusters 3, 7, 9, 10, 11 seem to all have very mixed images.

There are many possible reasons why not all results are not that satisfactory:

[Model/Features]
MobileNetV2 is pre-trained on ImageNet with has 1,000 classes (https://deeplearning.cms.waikato.ac.nz/user-guide/class-maps/IMAGENET/), if my dataset contains very random (which seems to be the case) images and is very different from those ImageNet classes, the image clustering might not be that effective.

Maybe choose other models?

[K number]
The chosen k number might not be the optimal one (the elbow plot also doesn't really show a clear elbow)

Maybe choose an alternative method of finding k numbers?

But also to note that no clearly defined optimal k number doesn't mean there is no cluster in the data.

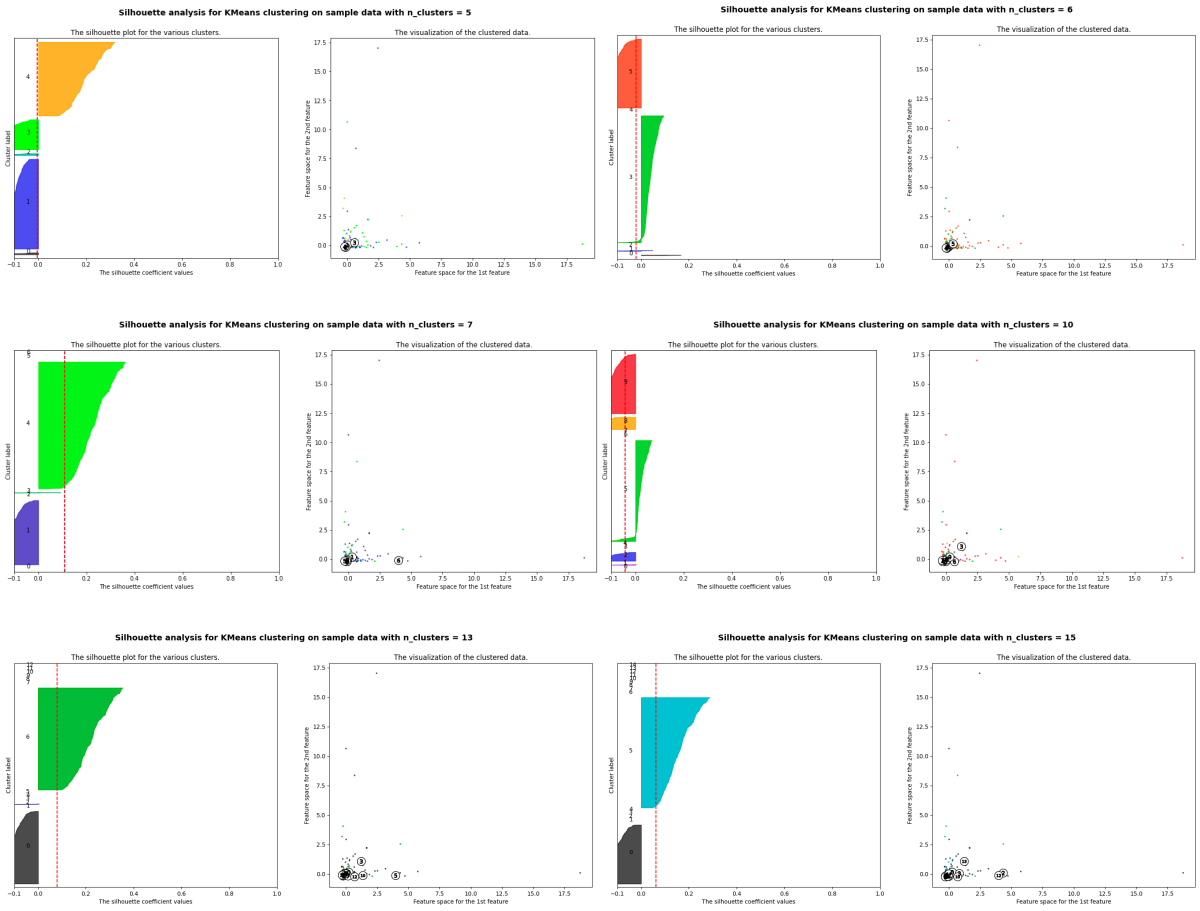
[Dataset]
The dataset is very random (randomly downloaded using Unsplash API) and imbalanced in terms of the category, and k-means struggle with clustering data where clusters have varying sizes (Patel, 2023).

There are also lots of abstract images that lack clear objects or themes in the dataset which might make the model harder to extract features.

It could also be a combination of several factors, here, to focus on trying to find the optimal k number – I tried another method to see if it could help me choose a better k number.

[Silhouette Diagrams]

According to Tomar (2023), the silhouette score/diagram can be useful for determining the optimal number of clusters when the elbow plot does not show a clear elbow point. I have plotted diagrams using the code adapted from - scikit-learn.org - selecting the number of clusters with silhouette analysis on KMeans clustering and ChatGPT of code debugging. The below are some of the example diagrams (n_cluster = 2, 5, 7, 10), which all seem to be performing poorly and can't really help me in finding a suitable k number.



For the tested numbers of clusters, many results show that the clusters have below-average silhouette scores or even less than 0, indicating a significant amount of mislabelled data or overlapping clusters. The dataset – as I mentioned before, is very random, so it is possible that it does contain many overlapping classes, making it challenging to do a clear image clustering. Additionally, it could also be that the current feature extraction method/model (imagenet) is not capturing the correct information, or it is challenging to do so (there are many abstract images in the dataset).

Compared silhouette diagrams with the elbow method:
- the elbow plot is more straightforward and intuitive in selecting the k-number, however, it doesn't really show the structure of the data, depending on the dataset, it may also not be able to provide a clear 'elbow' point.
- silhouette scores/diagrams provide more information than the elbow plot, providing a detailed view, which can be valuable for looking into complex datasets.

Overall, while the clustering results are somewhat meaningful, it is clear that this dataset is imbalanced and difficult to find an optimal k-number. K-means might not be suitable for this kind of dataset. Other algorithms, such as DBSCAN or Gaussian Mixture Models might be worth exploring as they can better handle clusters of arbitrary shapes (Patel, 2023).

5 Code, Supporting Document, and LLM Disclaimer

All datasets, code, Jupyter Notebooks, and clustering results (images) are available to access via the GitHub repository: https://git.arts.ac.uk/23001934/ds-portfolio / https://git.arts.ac.uk/23001934/ds-portfolio-weather-cat.

ChatGPT 3.5 has been used in this exercise for some proofreading, code debugging and for searching certain Python codes which have not been taught in class (especially for the part of silhouette analysis and creating webpage using javascript).

6 References and External Resources

References:

[1] Google Developers (2019). *k-Means Advantages and Disadvantages* . [online] Google Developers. Available at: https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages.

[2] Patel, K. (2023). *Understanding the Limitations of K-Means Clustering.* [online] Medium. Available at: https://medium.com/@kadambaripatel79/understanding-the-limitations-of-k-means-clustering-1fb5335f7859.

[3] Tomar, A. (2023). *Stop Using Elbow Method in K-means Clustering*. [online] builtin.com. Available at: https://builtin.com/data-science/elbow-method.

External Resources (Blogs, Posts, etc.):

[1] https://www.geeksforgeeks.org/python-pandas-dataframe-at/

[2] https://stackoverflow.com/questions/17011357/what-is-the-difference-between-content-and-text

[3] https://blog.apify.com/save-image-python/

[4] https://www.udemy.com/course/100-days-of-code/learn/lecture/22060308#overview

[5] https://stackoverflow.com/questions/51527868/how-do-i-embed-a-gif-in-jupyter-notebook

[6] https://builtin.com/data-science/elbow-method

[7] https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

[8] https://regex101.com/

[9] https://html5up.net/

**Detailed annotations were included in the Jupyter Notebook along with the code to indicate where ChatGPT and external resources, such as StackOverflow posts and other Python study materials were used.*