



DATA SCIENCE

Join the Flickr community, home to tens of billions of photos and 2 million groups.

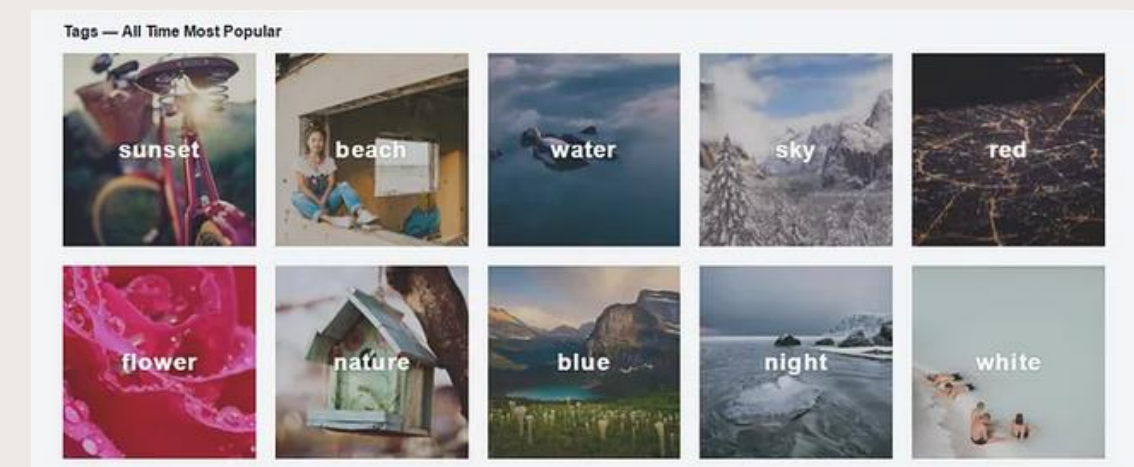
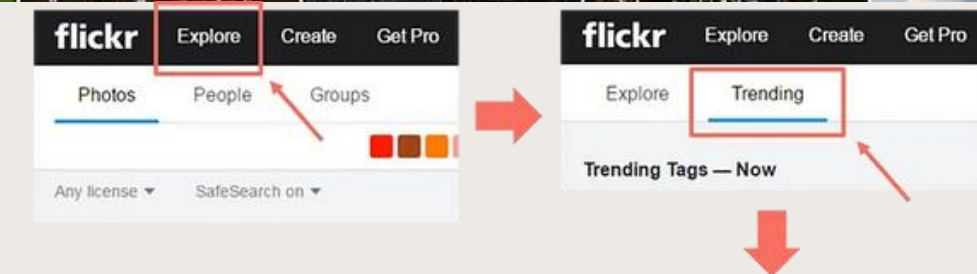
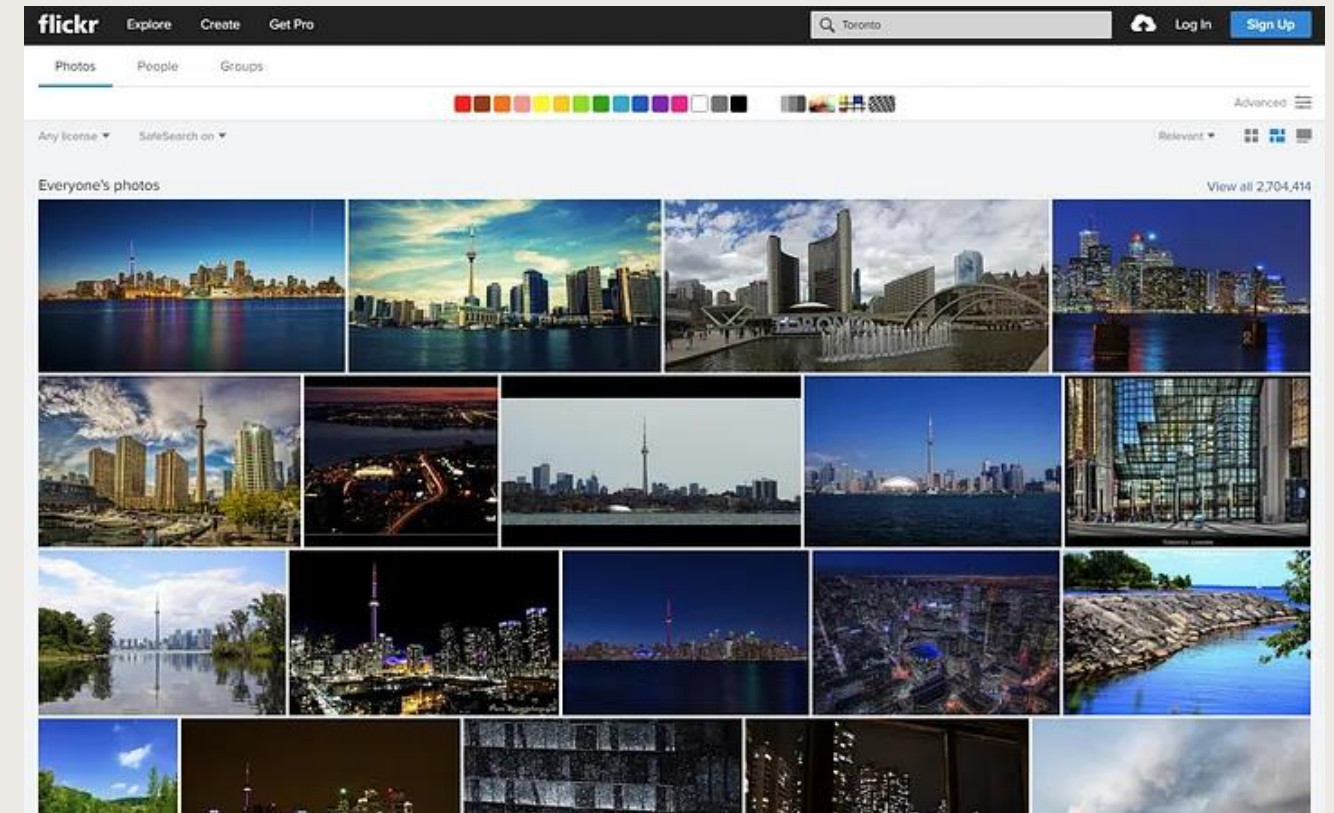
Bar Harush
Matan Katsnelson



About flickr

- Flickr is a social photo-sharing platform created in 2004.
- It emphasizes community engagement through features like comments, likes, and groups.
- Flickr allows users to connect, interact, and share their photos with others who have similar interests.

THANKS TO WIKIPEDIA.



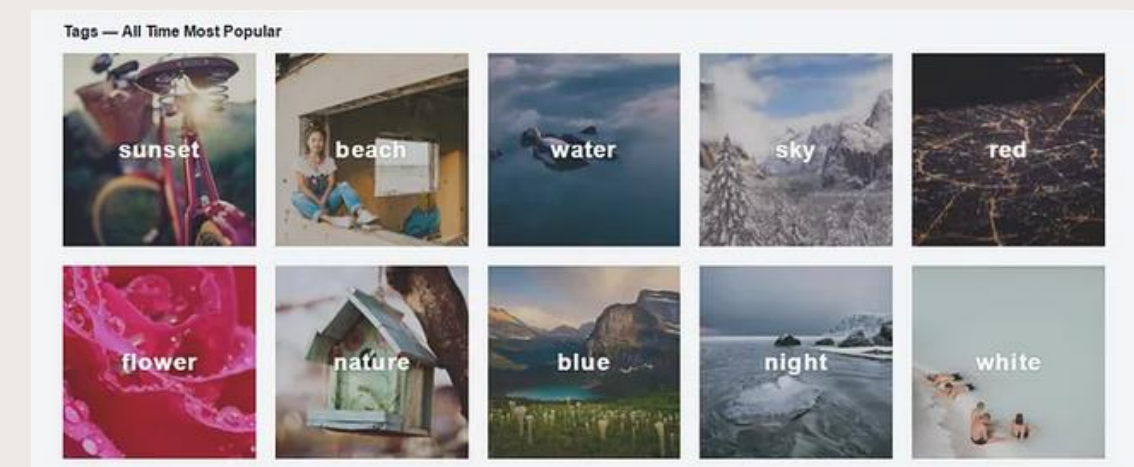
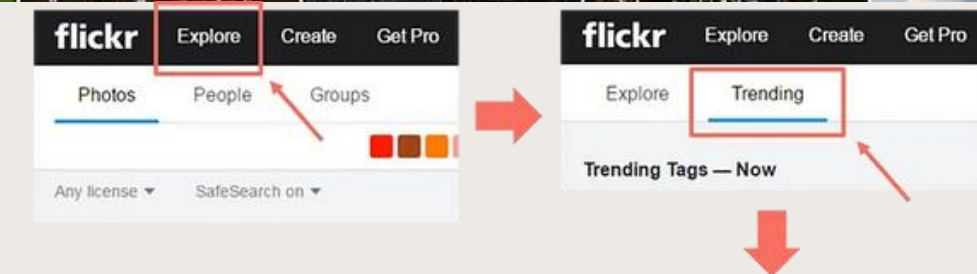
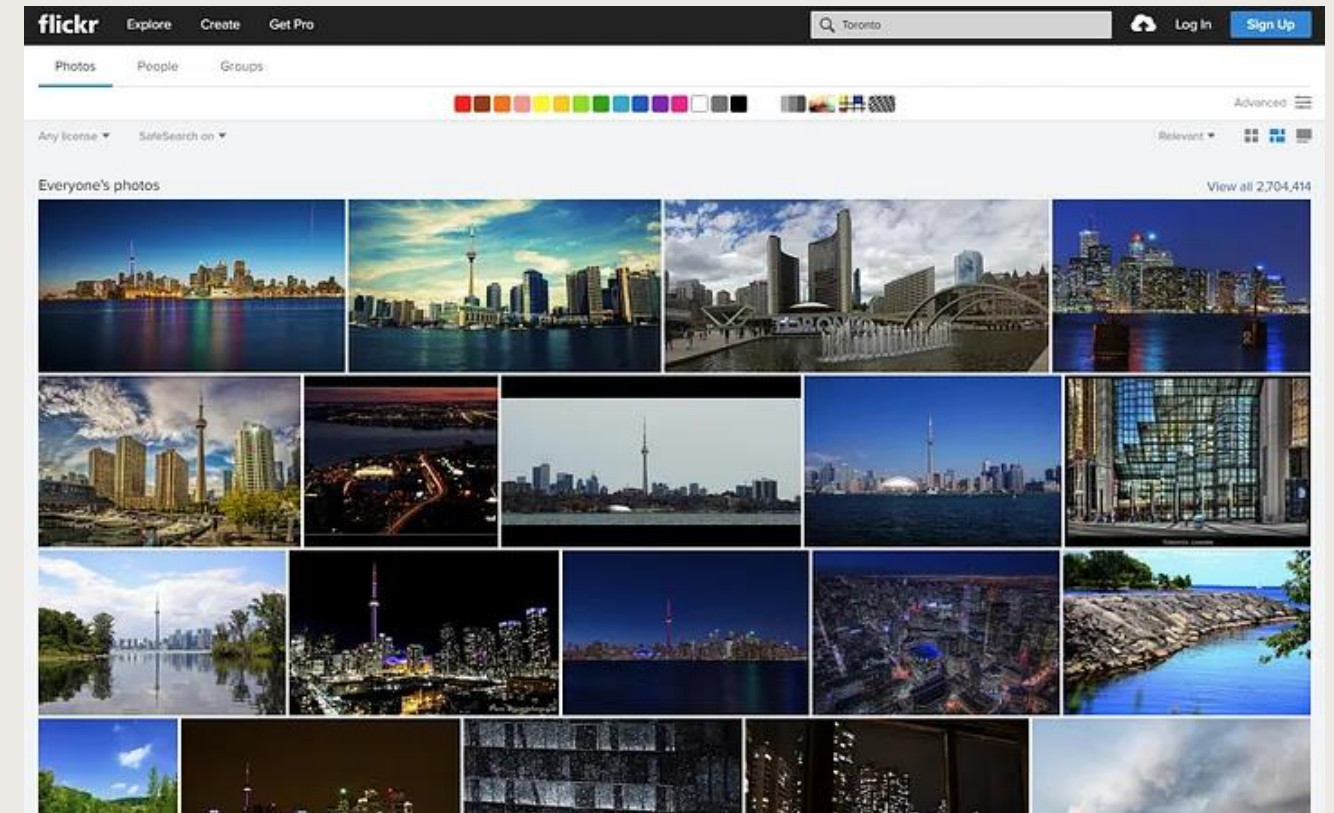
[THANKS TO TIANHAO WU.](#)



About flickr

- Flickr is a social photo-sharing platform created in 2004.
- It emphasizes community engagement through features like comments, likes, and groups.
- Flickr allows users to connect, interact, and share their photos with others who have similar interests.

THANKS TO WIKIPEDIA.

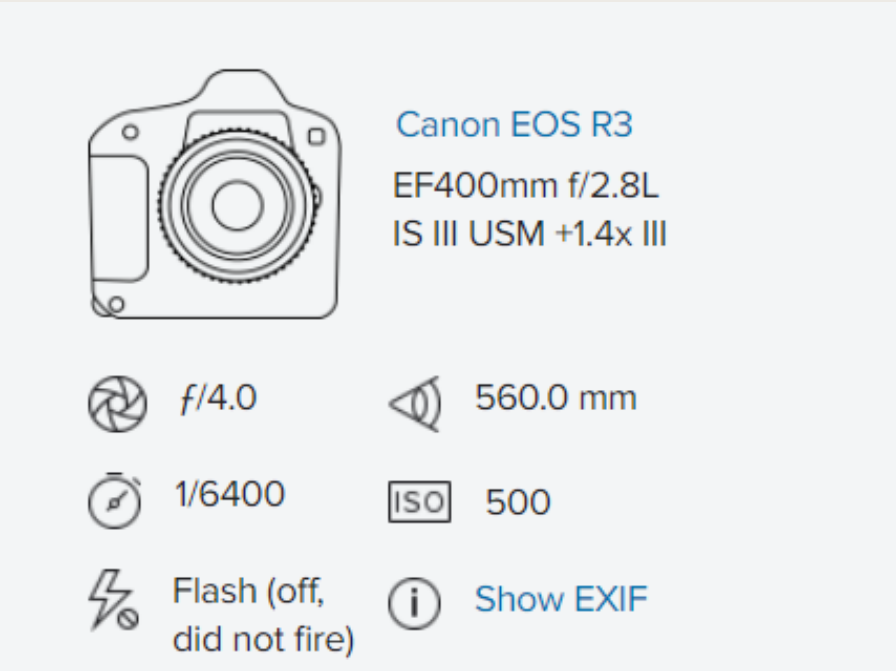
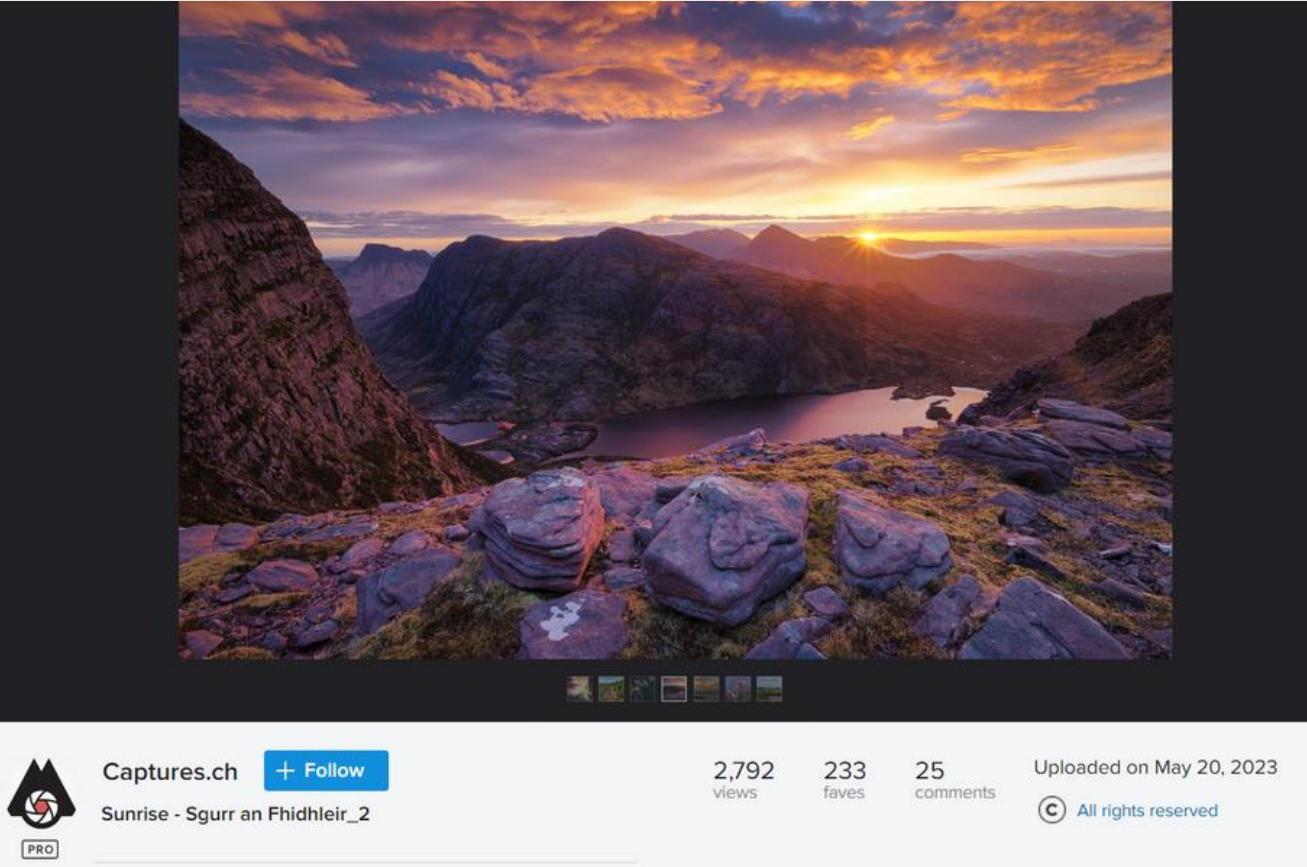
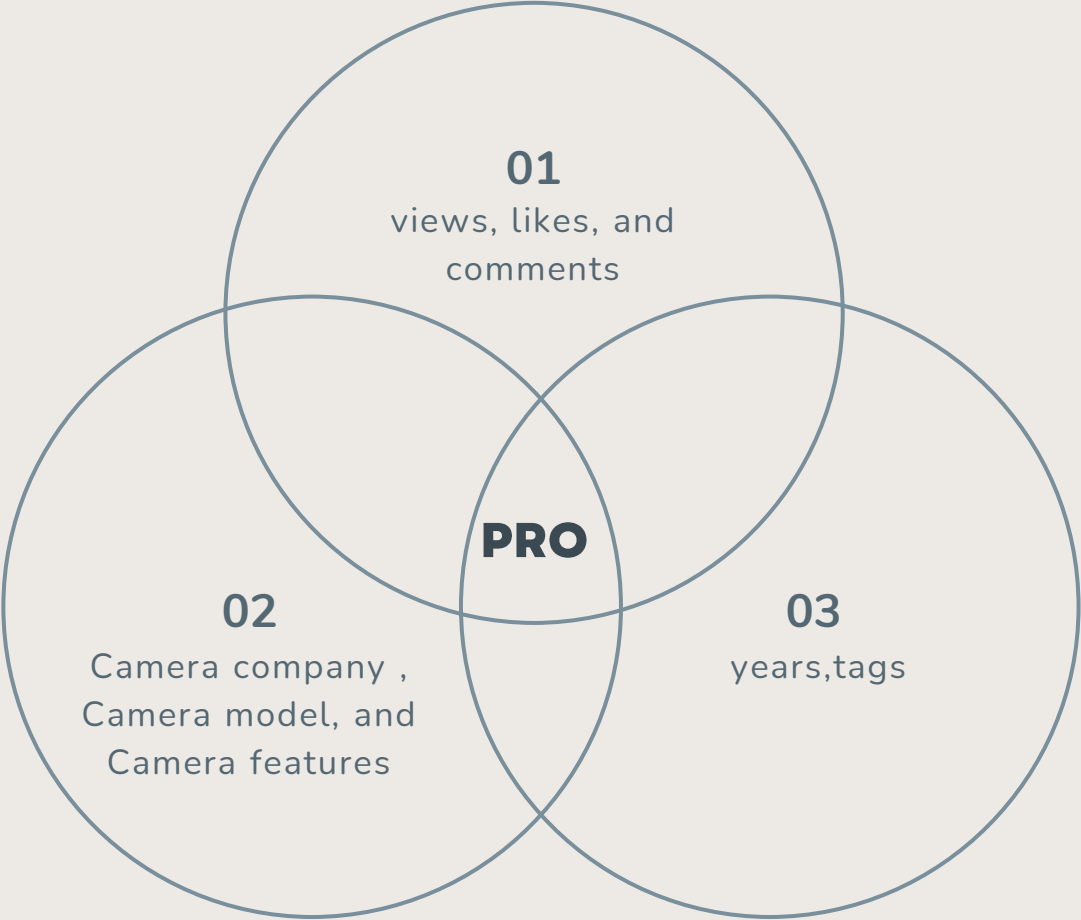


[THANKS TO TIANHAO WU.](#)



CAN "PRO" BE PREDICTED ON THE FLICKR WEBSITE?

- Is there a correlation between views, likes, and comments on "Pro"?
- Is there a correlation between Camera company , Camera model, and Camera features on "Pro"?
- Is there a correlation between years, tags and "Pro"?



THE STAGES IN THE PROJECT

1

Scraping and Crawling

2

Data Cleaning

3

Eda-Visualization

4

A comprehensive study
on the website Flickr
and plan B

5

Machine Learning

6

Summary and Conclusions

1

SCRAPING AND CRAWLING

SELENIUM

FOR PHOTOS
PER TAGSFOR CAMERA
features

BEAUTIFULSOUP

FOR GENERAL
INFORMATION

```
url = f"https://www.flickr.com/photos/tags/{search_term}/page{i}"
```

< 1 2 ... 74 75 76 77 78 79 80 >



```
from selenium import webdriver
from selenium.webdriver.firefox.options import Options as FirefoxOptions
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities
import time
from bs4 import BeautifulSoup
import csv

# FIREFOX PATHS AND OPTIONS
filename = 'flickr_urls.csv'
driver_path = r"C:\Users\barex\Downloads\geckodriver-v0.33.0-win32\geckodriver.exe"
firefox_binary = 'C:\Program Files\Mozilla Firefox\Firefox.exe'

firefox_options = FirefoxOptions()
firefox_options.binary_location = firefox_binary
firefox_options.add_argument('--auto-audio')
firefox_options.add_argument('--headless')

capabilities = DesiredCapabilities.FIREFOX.copy()
capabilities['acceptInsecureCerts'] = True

search_terms = ("arizona": "arizona", "nature": "nature", "purple": "purple",
               "art": "art", "bridge": "bridge", "shop": "shop",
               "farm": "farm", "bar": "bar", "day": "day", "house": "house",
               "storage": "storage", "flowers": "flowers", "car": "car",
               "dog": "dog", "garden": "garden", "music": "music",
               "sky": "sky", "snow": "snow", "river": "river", "sun": "sun",
               "sunset": "sunset", "beach": "beach", "cat": "cat")

with webdriver.Firefox(executable_path=driver_path, options=firefox_options, desired_capabilities=capabilities) as driver:
    all_urls = []

    for search_term, tag in search_terms.items():
        urls = []
        for i in range(1, 81):
            url = f"https://www.flickr.com/photos/tags/{search_term}/page{i}"
            driver.get(url)
            time.sleep(2)
            if driver.current_url != url:
                break
            SCROLL_PAUSE_TIME = 1
            last_height = driver.execute_script("return document.body.scrollHeight")
            count = 0
            while True:
                driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
                time.sleep(SCROLL_PAUSE_TIME)
                try:
                    load_more = driver.find_element_by_xpath("//button[@class='load-more-suggestions loaded-suggestions-show']")
                    load_more.click()
                    time.sleep(SCROLL_PAUSE_TIME)
                except:
                    break
                if driver.find_elements_by_xpath("//a[@class='page-arrow right']") == []:
                    break
                count += 1
                if count >= 10000:
                    break
                new_height = driver.execute_script("return document.body.scrollHeight")
                if new_height == last_height:
                    break
                last_height = new_height
            html = driver.page_source
            soup = BeautifulSoup(html, 'html.parser')
            links = []
            for a in soup.select('div.view.tag-photos-everyone-view a.overlay'):
                link = a.get('href')
                links.append(f'https://www.flickr.com{link}')
            all_urls.setdefault(search_term, []).extend(links)
            if len(links) == 0:
                break

    driver.quit()

    for tag, urls in all_urls.items():
        print(f"len(urls) {len(urls)} urls found for {tag}")

    with open(filename, mode='w', newline='') as csv_file:
        writer = csv.writer(csv_file)
        writer.writerow(['search term', 'image url'])

    for tag, urls in all_urls.items():
        for url in urls:
            writer.writerow([tag, url])
    image_pageimage.pngimage.png
    print("Data saved to {filename}")
```

```
def get_full_page(url):
    driver_path = r"C:\Users\barex\Downloads\geckodriver-v0.33.0-win32\geckodriver.exe"
    firefox_binary = r"C:\Program Files\Mozilla Firefox\Firefox.exe"

    options = Options()
    options.headless = True
    driver = webdriver.Firefox(options=options, firefox_binary=firefox_binary, executable_path=driver_path)

    driver.get(url)
    elem = driver.find_element_by_tag_name("body")

    # Scroll down to the bottom of the page
    no_of_pagedowns = 2
    for i in range(no_of_pagedowns):
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(0.5)
    # Wait for the page to load completely
    full_page = driver.page_source
    driver.quit()

    return full_page

def get_info_dict(url):
    info = {
        'exif-camera-name': 'camera name',
        'c-charm-item-aperture': 'Aperture',
        'c-charm-item-focal-length': 'Focal length',
        'c-charm-item-exposure-time': 'Exposure time',
        'c-charm-item-flash': 'Flash',
        'location-name-link': 'location'
    }

    i = 1
    i += 1

    info_dict = {'url': url}

    try:
        full_page = get_full_page(url)
        soup = BeautifulSoup(full_page, 'html.parser')

        for class_name, key in info.items():
            value = soup.find(class_=class_name)
            if value:
                value = value.text.strip()
                if value == 'NaN':
                    info_dict = {k: 'NaN' for k in info.values()}
                    break
            else:
                value = 'NaN'
                info_dict[key] = value

    except Exception as e:
        if i % 100 == 0:
            save_csv(results)
            print(f"the index is '{i}' saved csv")

        # Wait for 1 second between requests
        time.sleep(1)

    return info_dict
```

```
def scrape_camera_data(urls):
    headers = ['url', 'camera name', 'Aperture', 'Focal length', 'Exposure time', 'Flash', 'location']

    with concurrent.futures.ThreadPoolExecutor() as executor:
        for j in range(0, len(urls), 10):
            results = executor.map(get_info_dict, urls[j:j+10])

            # Convert generator object to list
            results = list(results)

            save_csv(results, headers)

            if j % 1000 == 0:
                print(f"Saved csv at index: {j}")

            # Wait for 1 second between requests
            time.sleep(1)

    return results

def save_csv(results, headers):
    filename = 'final_camera_info_check.csv'

    # Check if the file exists and add header if it doesn't
    if not os.path.exists(filename):
        with open(filename, mode='w', newline='', encoding='utf-8') as file:
            writer = csv.writer(file)
            writer.writerow(headers)

    # Append data to the csv file
    with open(filename, mode='a', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        for result in results:
            writer.writerow([result.get('url', ''),
                            result.get('exif-camera-name', ''),
                            result.get('Aperture', ''),
                            result.get('Focal length', ''),
                            result.get('Exposure time', ''),
                            result.get('Flash', ''),
                            result.get('location', '')])
```

```
def scrape_links_General_information(urls, tags):
    class_to_key = {
        'view-count-label': 'view',
        'fave-count-label': 'likes',
        'comment-count-label': 'comments',
        'date-taken-label': 'date',
        'icon-pro-badge': 'pro',
        'owner-name': 'photographer'
    }

    new_urls = set()
    for url in urls:
        if '%20' in url:
            new_url = url.replace('%20', ' ')
            new_urls.add(new_url)
        else:
            new_urls.add(url)

    scraped_data = []
    j=1
    i=1
    for url, tag in zip(new_urls, tags):
        # Check if the URL starts with "http://" or "https://"
        if not url.startswith(("http://", "https://")):
            url = "http://" + url
        info_dict = {'url': url, 'tag': tag}

        if((i==j)):
            print(i)
            j=j+10
            i=i+1
        else:
            i=i+1

        soup = BeautifulSoup(requests.get(url).content, 'html.parser')
        for class_name, key in class_to_key.items():
            value = soup.find(class_=class_name)
            if value is not None:
                if key == 'pro':
                    value = '1'

                elif key == 'photographer':
                    value = value.text.strip()
                elif key == 'date':
                    value = value.text.replace('Taken on ', '').strip()
                else:
                    value = value.text.strip()

            else:
                if key == 'pro':
                    value = '0'
                else:
                    value = 'NaN'
            info_dict[key] = value

        scraped_data.append(info_dict)

    return scraped_data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73344 entries, 0 to 73343
Data columns (total 18 columns):
Unnamed: 0      73344 non-null int64
Unnamed: 0.1    73344 non-null int64
url             73344 non-null object
tag             73344 non-null object
view           73297 non-null object
likes           73297 non-null object
comments       73297 non-null object
date           73297 non-null object
pro            73344 non-null int64
photographer    73297 non-null object
Camera model    49724 non-null object
Camera brand    49231 non-null object
camera name     49199 non-null object
Aperture        36810 non-null object
Focal length    73344 non-null object
Exposure time   73344 non-null object
Flash           36484 non-null object
location        12303 non-null object
dtypes: int64(3), object(15)
memory usage: 10.1+ MB
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 49176 entries, 0 to 49197
Data columns (total 18 columns):
Unnamed: 0      49176 non-null int64
Unnamed: 0.1    49176 non-null int64
url             49176 non-null object
tag             49176 non-null object
view           49176 non-null object
likes           49176 non-null object
comments       49176 non-null object
date           49176 non-null object
pro            49176 non-null int64
photographer    49176 non-null object
Camera model    49176 non-null object
Camera brand    49176 non-null object
camera name     49176 non-null object
Aperture        49176 non-null object
Focal length    49176 non-null object
Exposure time   49176 non-null object
Flash           49176 non-null object
location        49176 non-null object
dtypes: int64(3), object(15)
memory usage: 7.1+ MB
```

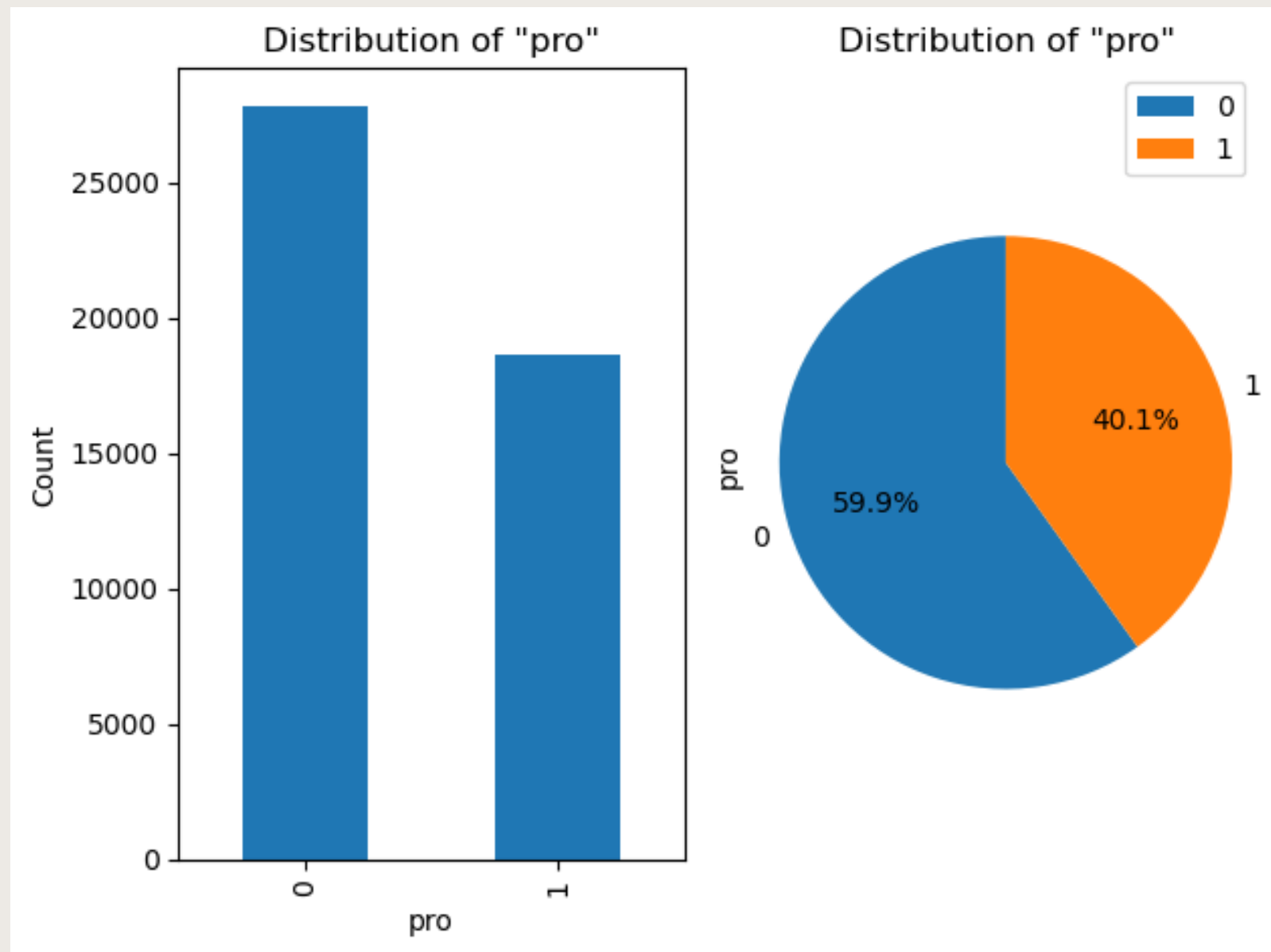
49198 rows × 17 columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46435 entries, 0 to 46434
Data columns (total 79 columns):
Unnamed: 0      46435 non-null int64
url             46435 non-null object
tag             46435 non-null object
view           46435 non-null int64
likes           46435 non-null int64
comments       46435 non-null int64
date           45936 non-null object
pro            46435 non-null int64
photographer    46435 non-null object
Camera model    46435 non-null object
Camera brand    46435 non-null object
camera name     46435 non-null object
Aperture        34614 non-null object
Focal length    46435 non-null float64
Exposure time   46435 non-null object
Flash           46435 non-null int64
location        8641 non-null object
date year       45936 non-null float64
more_pro_in_brand 46435 non-null float64
more_pro_in_model 46435 non-null float64
most pro in year 46435 non-null int64
Performances in Likes 46435 non-null float64
likes_views_ratio 46435 non-null float64
comments_views_ratio 46435 non-null float64
comments_likes_ratio 46435 non-null float64
Aperture Category 34420 non-null object
Focal Length Category 34375 non-null object
```

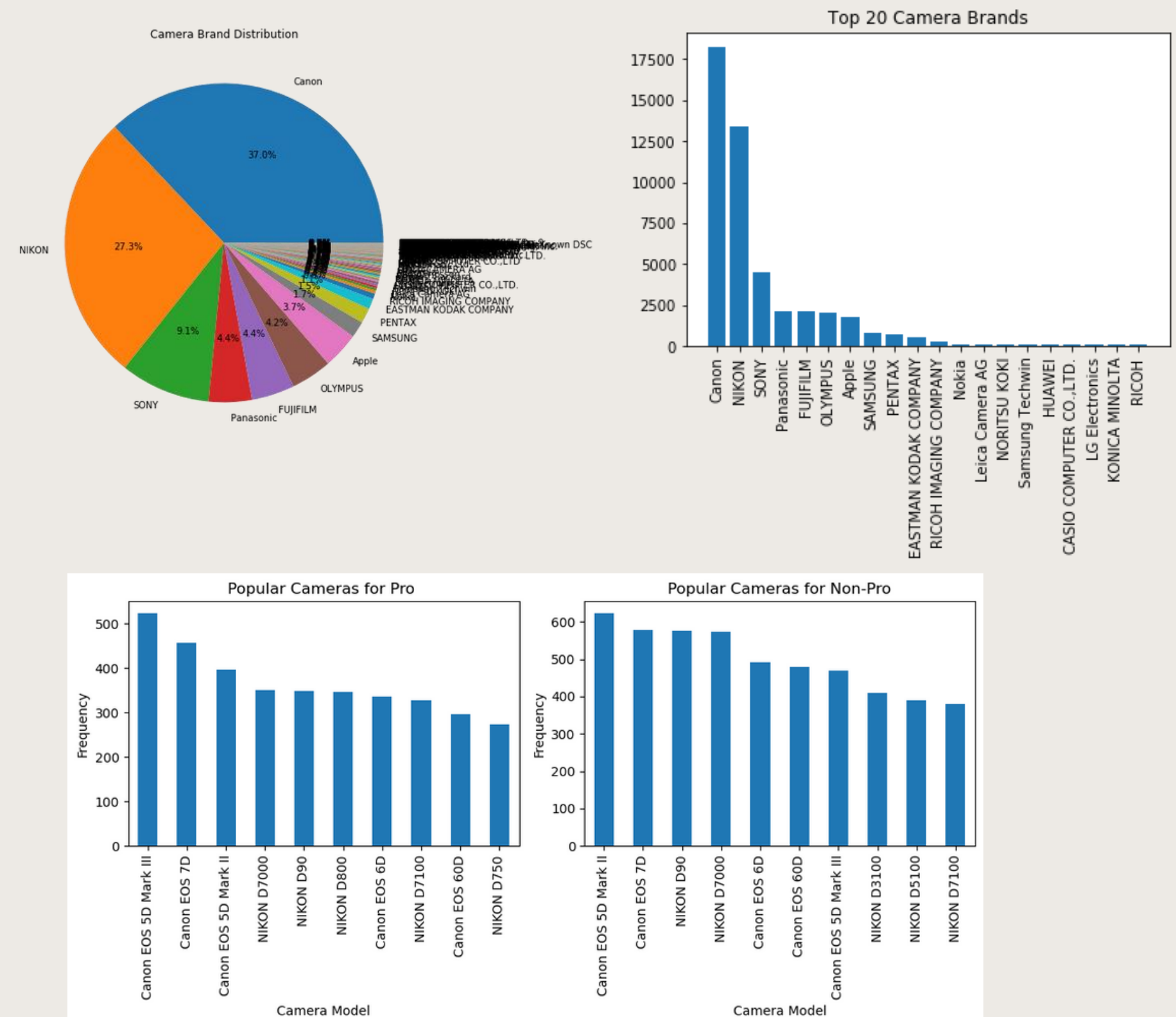
46435 rows × 79 columns

In the end, we reached from 73,000 to 46,000.

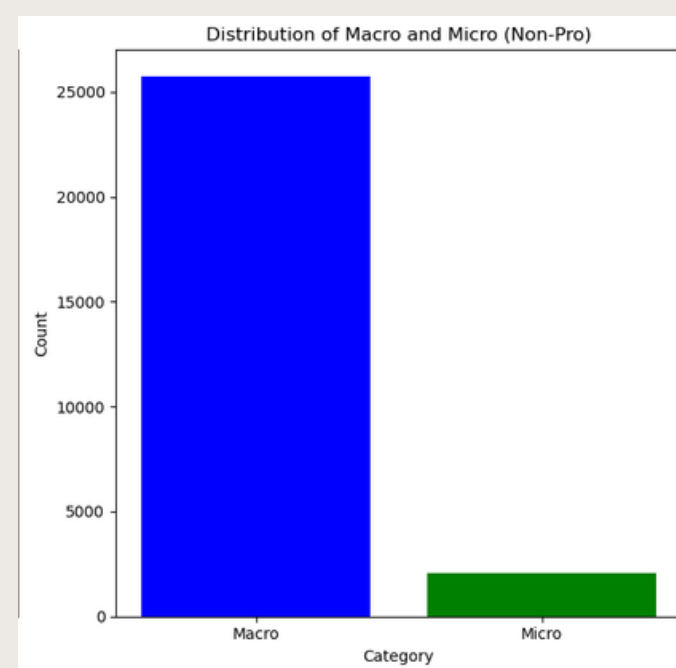
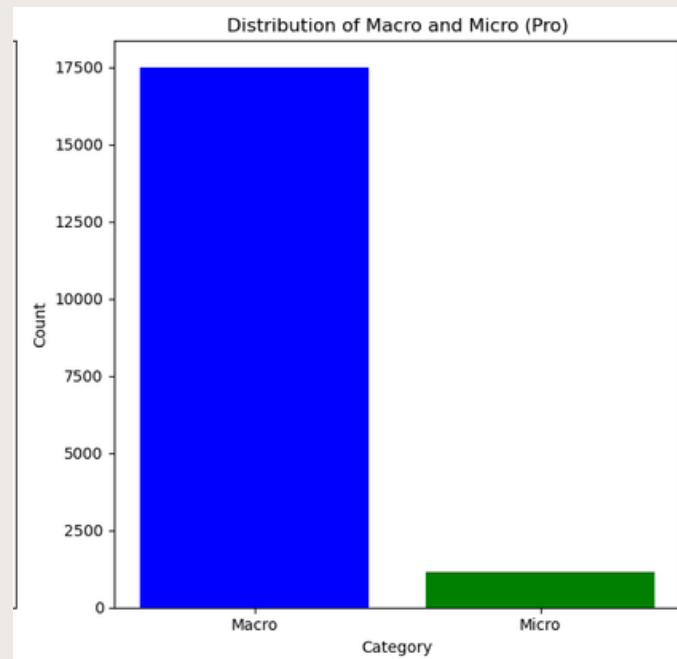
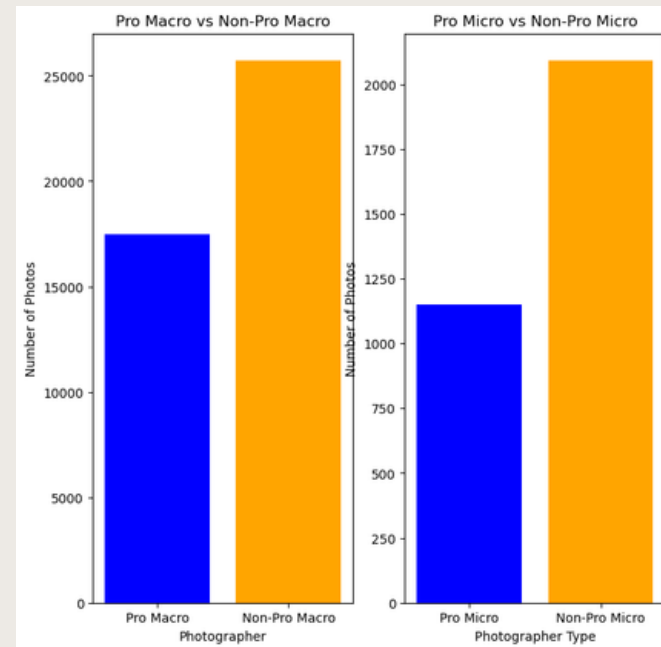
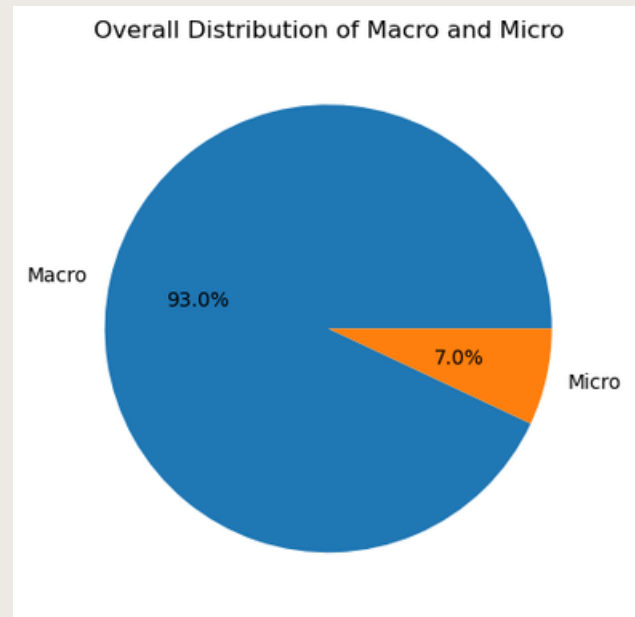
DISTRIBUTION PRO AND NON-PRO



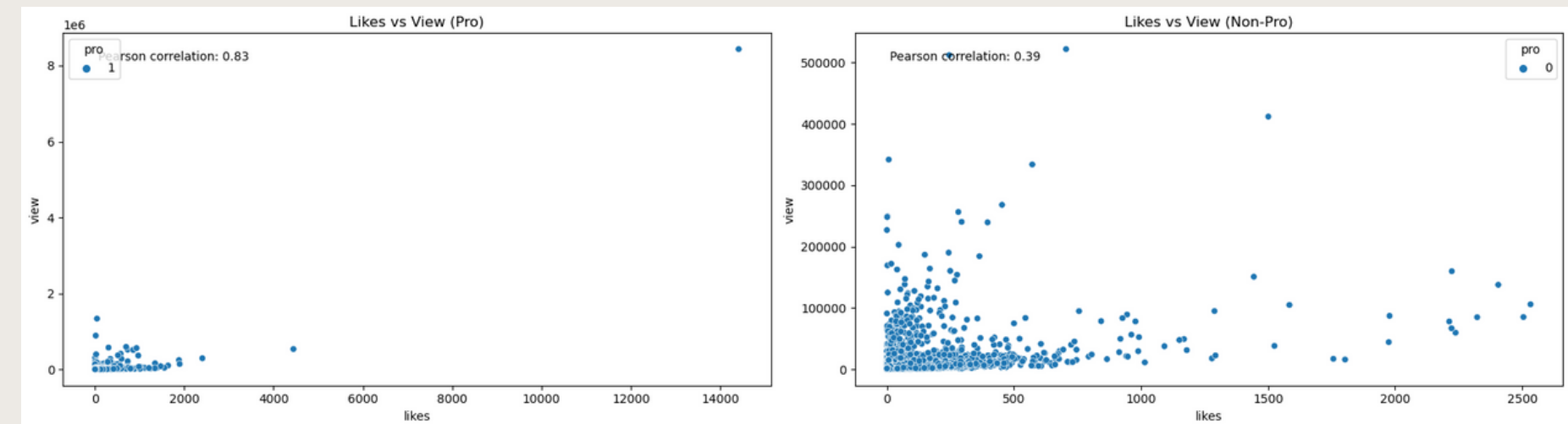
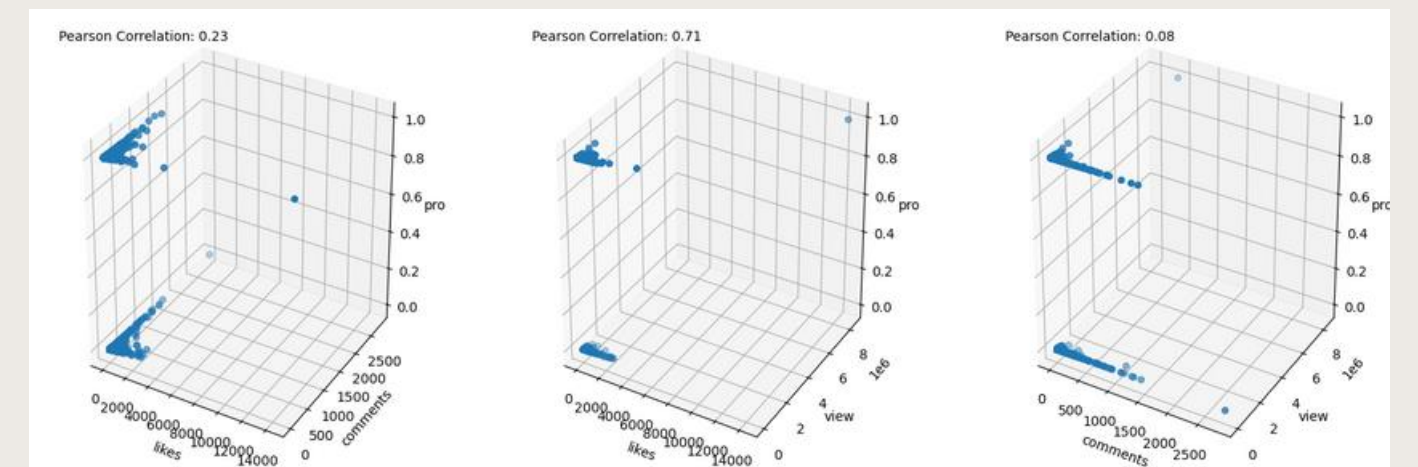
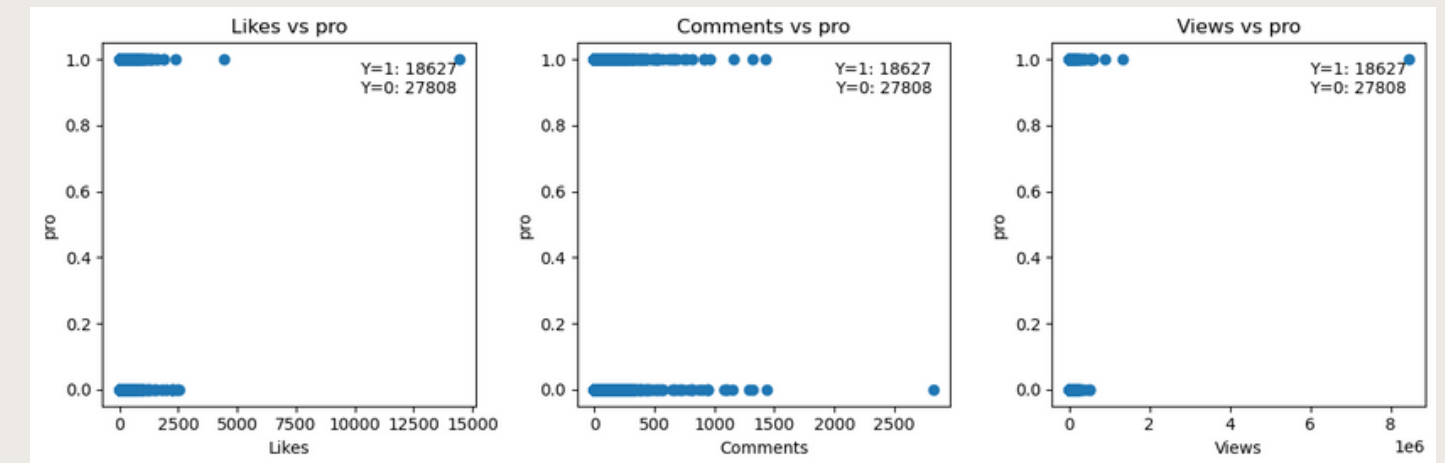
ATTEMPT TO FIND A CORRELATION BETWEEN THE TYPE OF CAMERA AND WHETHER IT IS PRO OR NOT.

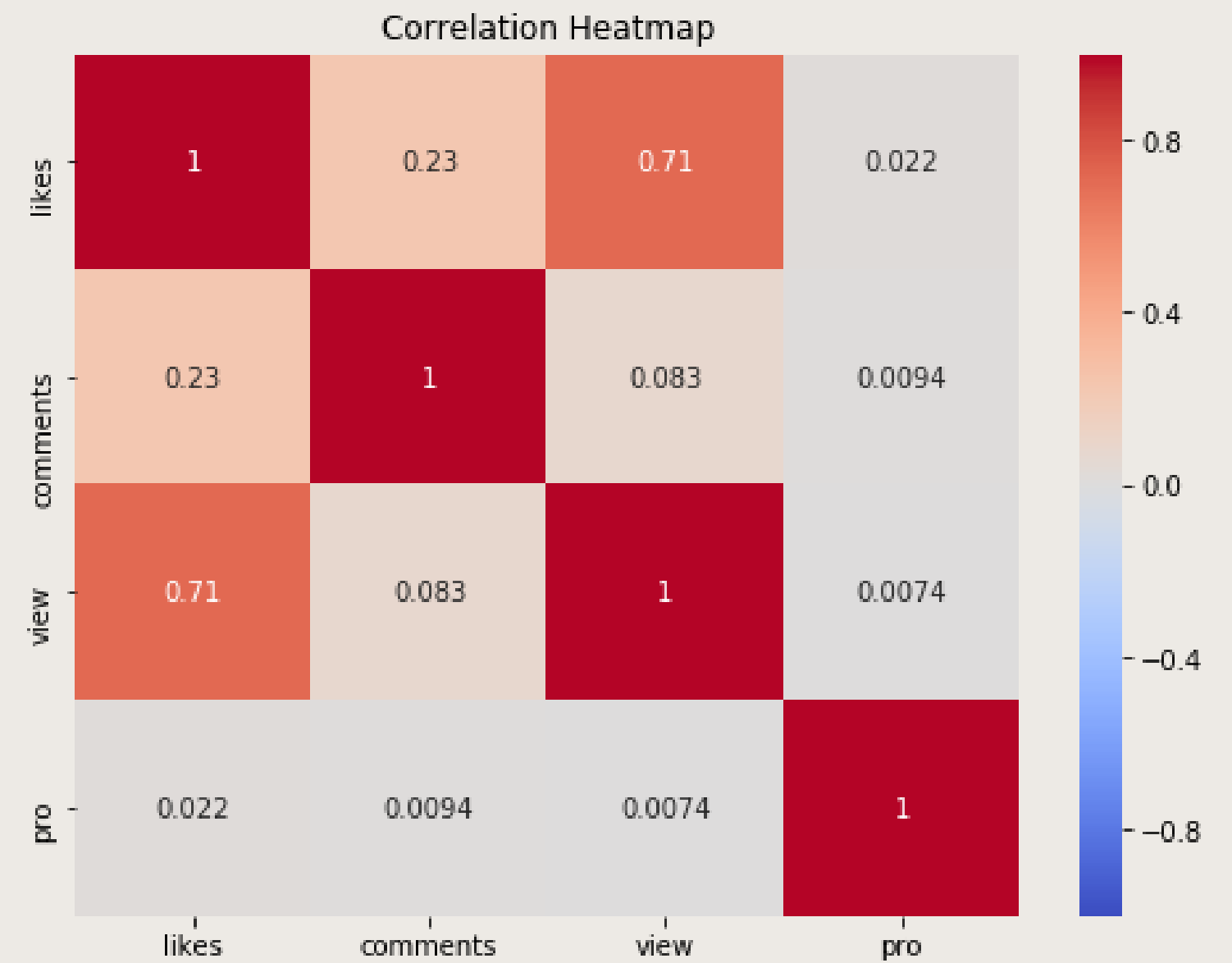
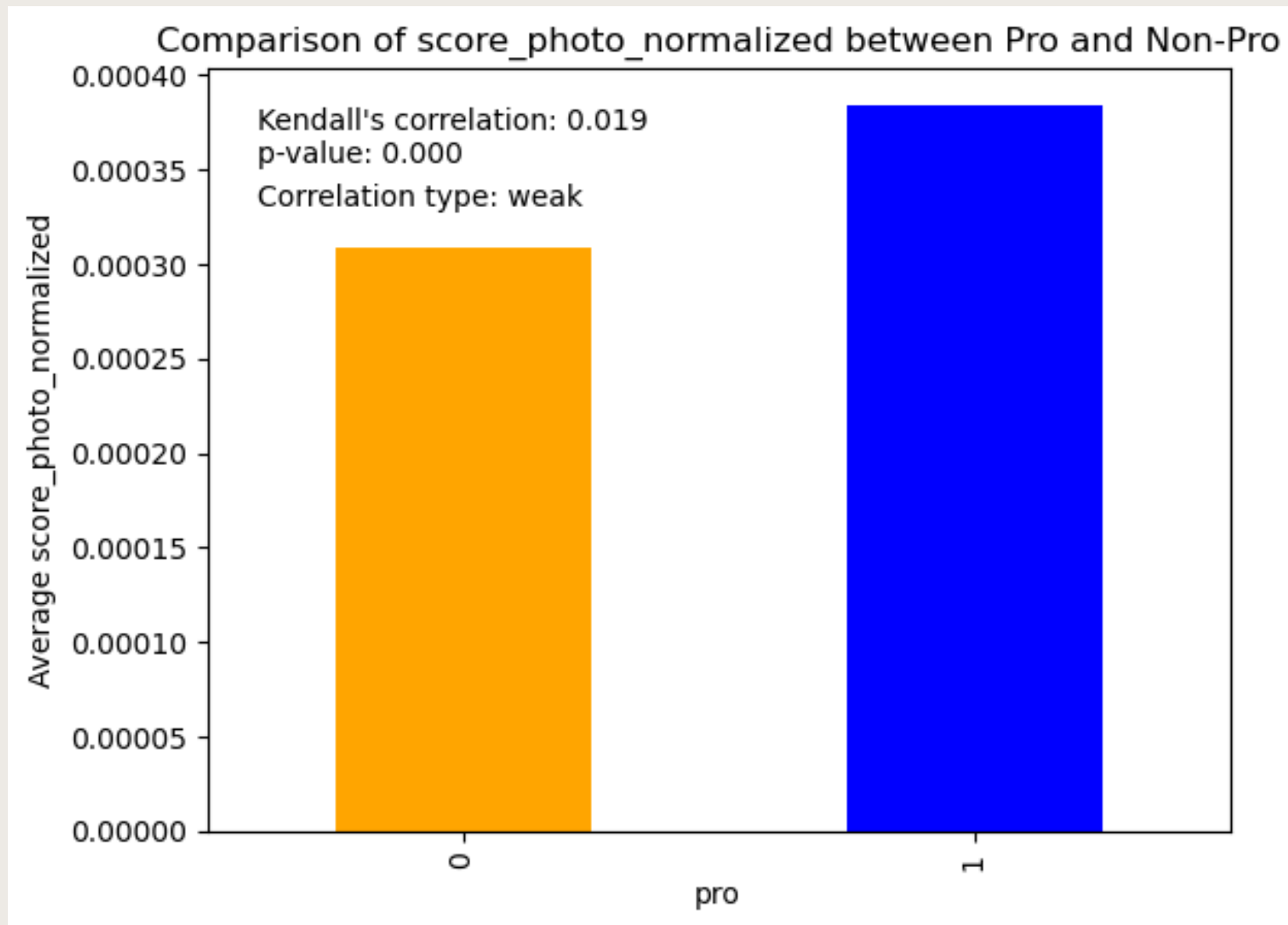


EXAMINING THE RELATIONSHIP BETWEEN PROFESSIONAL OR NON-PROFESSIONAL STATUS AND THE TYPE OF PHOTOGRAPHY.



ASSESSING THE CORRELATION BETWEEN LIKES, VIEWS, AND COMMENTS USING THE PEARSON CORRELATION COEFFICIENT AND EVALUATING ITS ASSOCIATION WITH THE 'PRO' ASPECT.

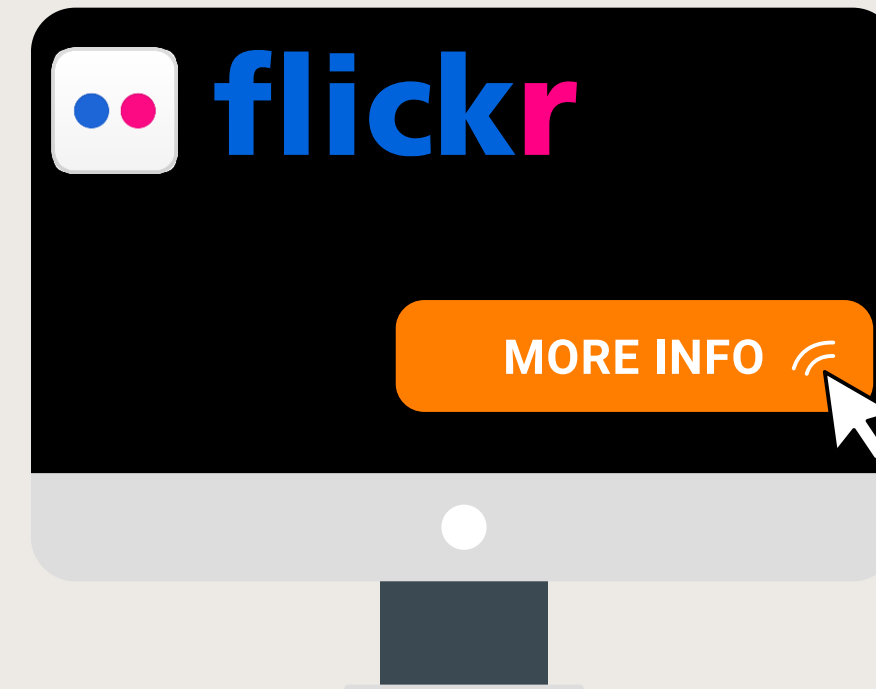
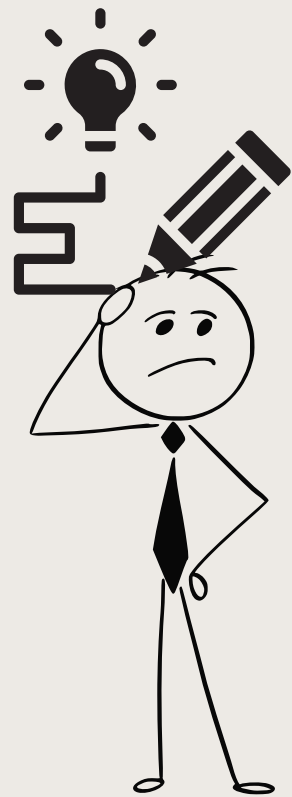




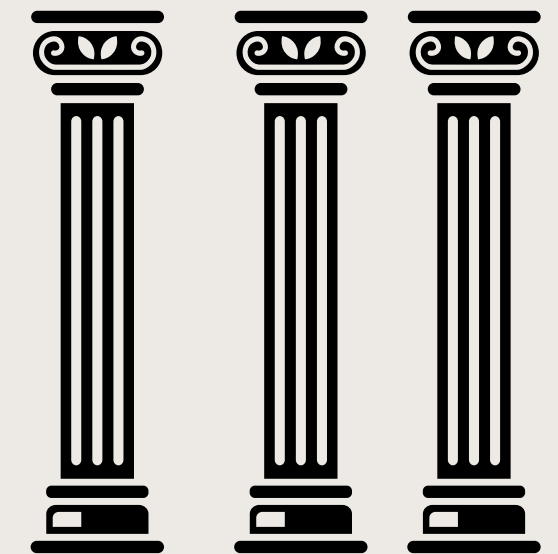
```
Correlation coefficient between 'likes' and 'pro': 0.021784754784267123
Correlation coefficient between 'comments' and 'pro': 0.00941584137871338
Correlation coefficient between 'views' and 'pro': 0.0073895582266121675
```


4

RESEARCH AND PLAN B

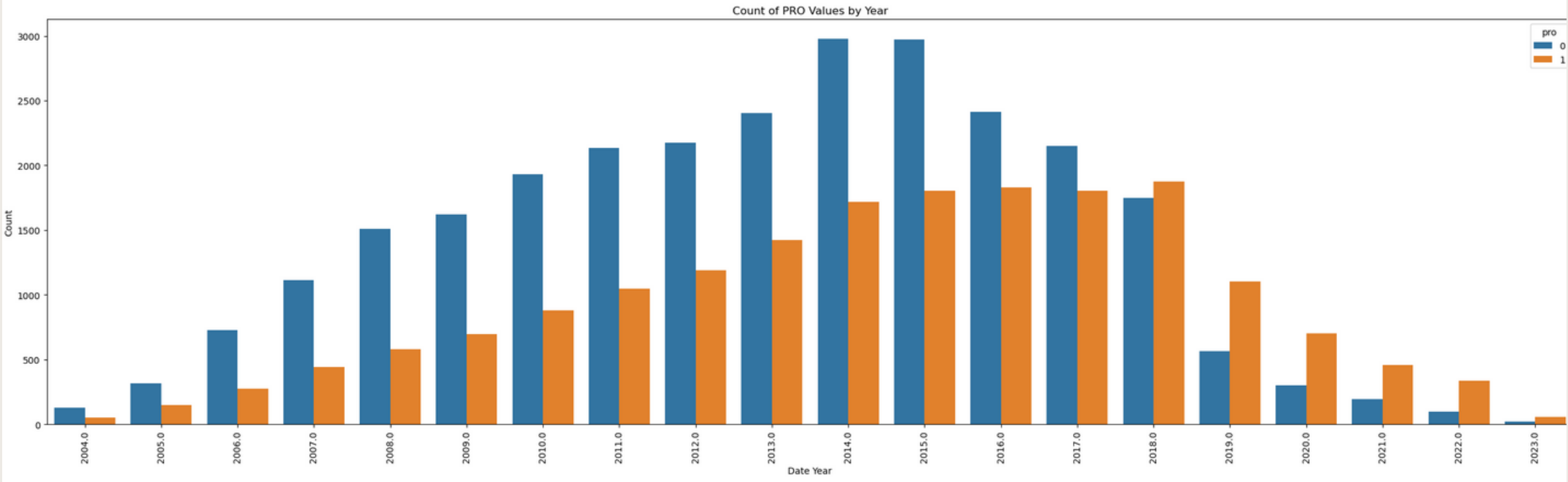


MORE COLUMN

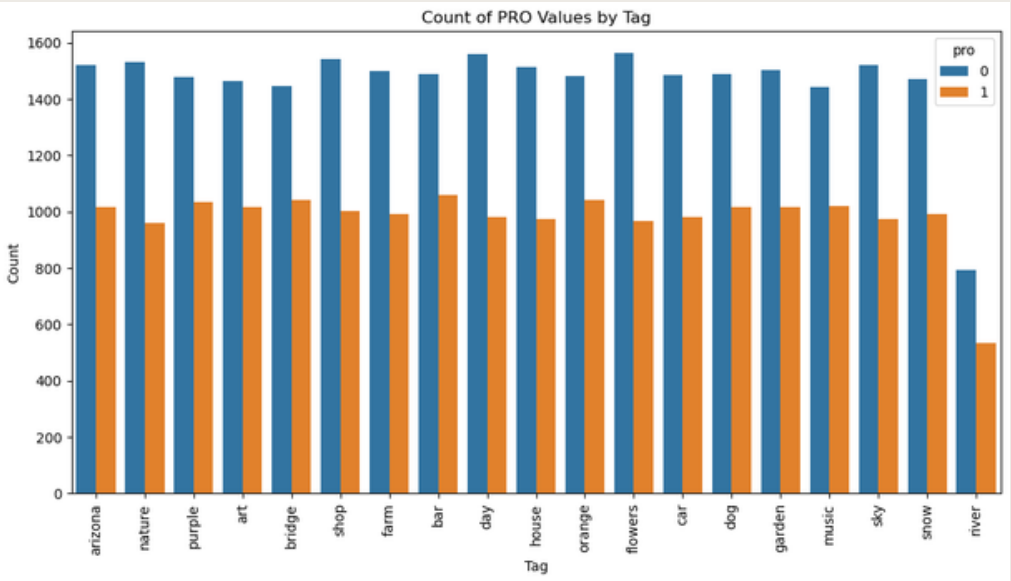


2018

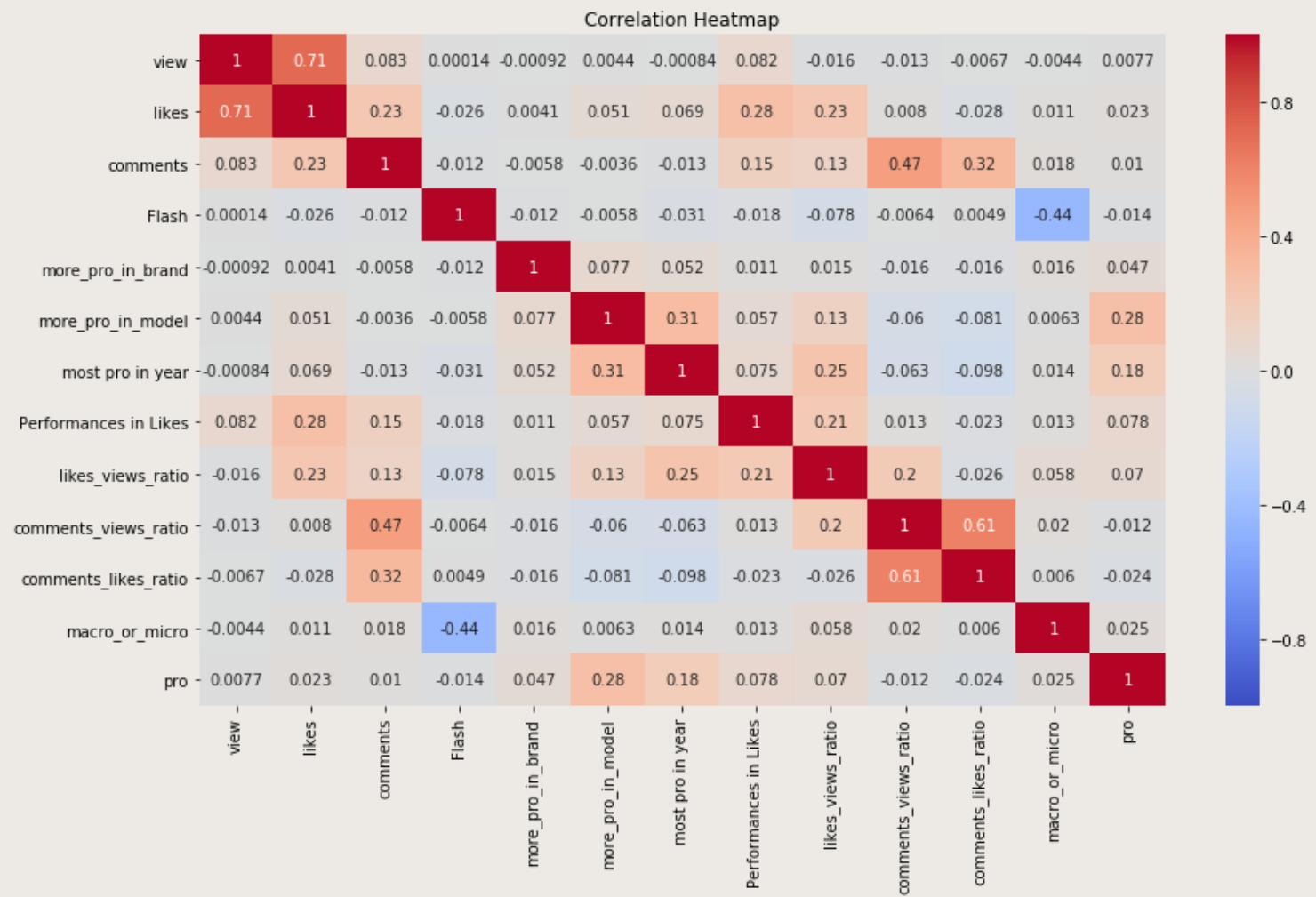
THE RELATIONSHIP BETWEEN YEARS AND "PRO" OR "NON-PRO" STATUS.



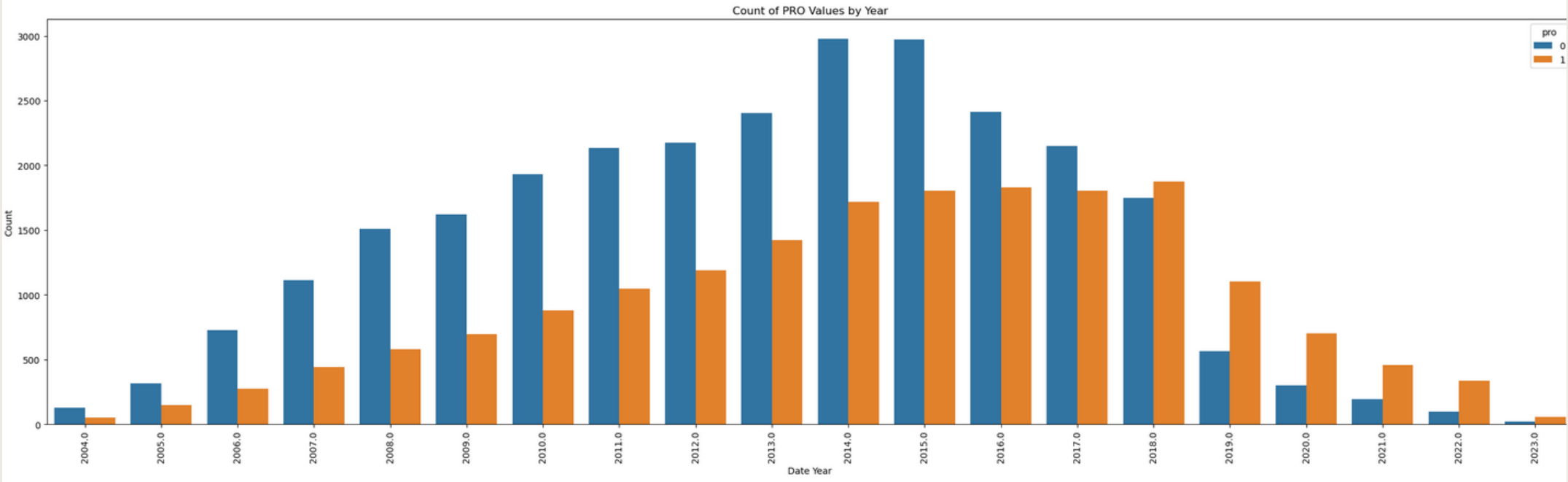
THE RELATIONSHIP BETWEEN TAGS AND "PRO" OR "NON-PRO" STATUS.



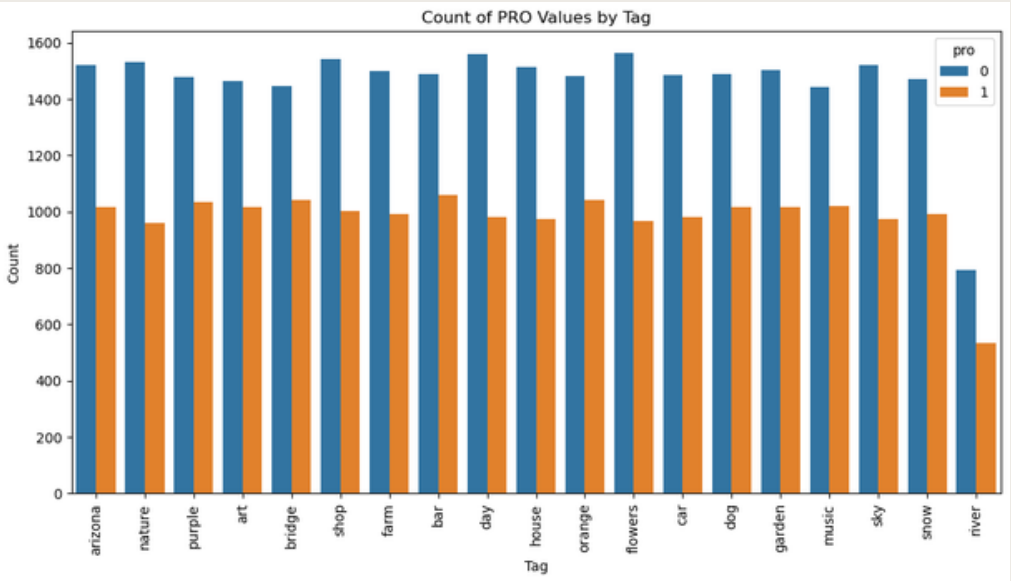
CHECK RELATIONSHIP WITH MORE COLUMNS IN HEATMAP.



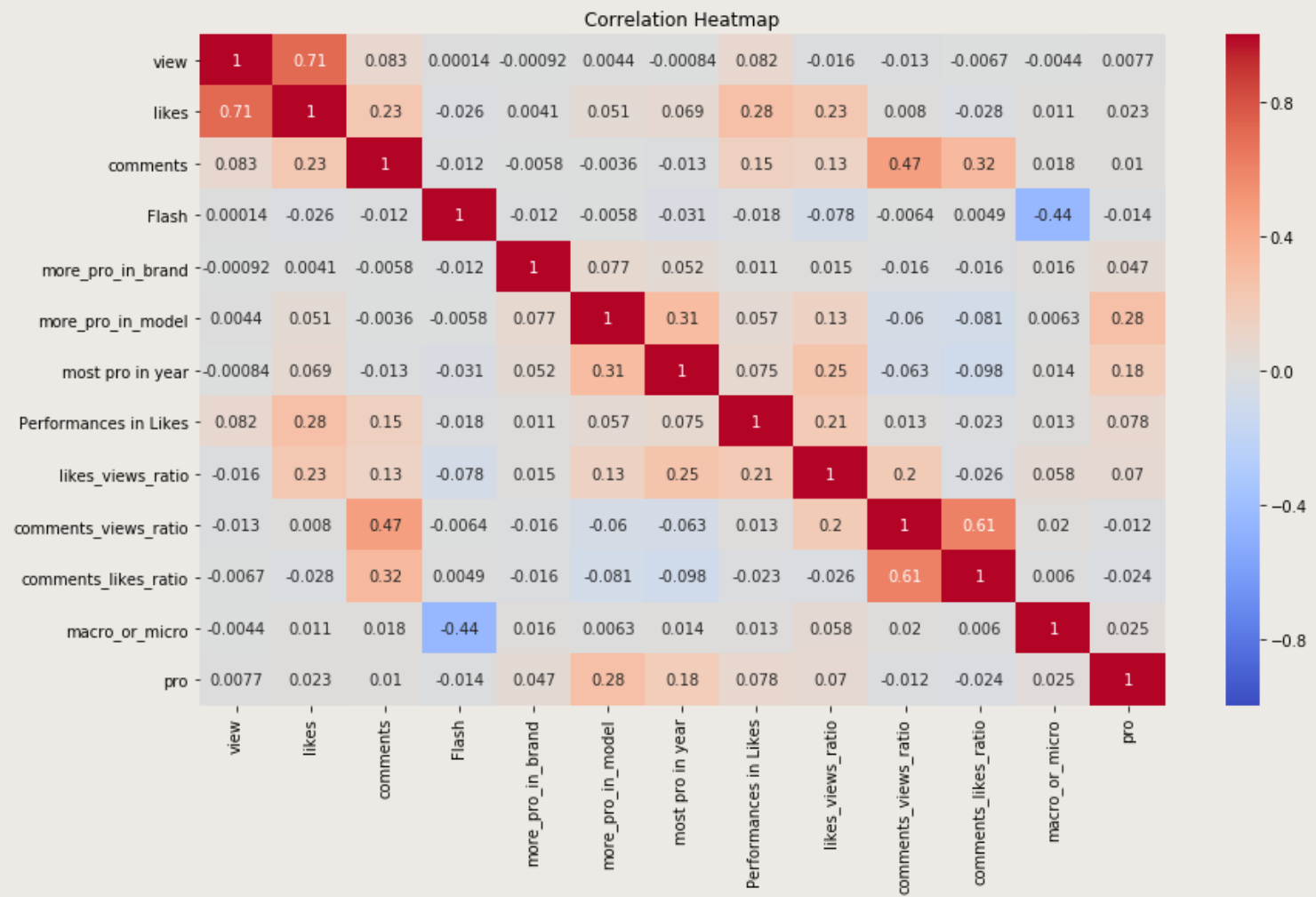
THE RELATIONSHIP BETWEEN YEARS AND "PRO" OR "NON-PRO" STATUS.



THE RELATIONSHIP BETWEEN TAGS AND "PRO" OR "NON-PRO" STATUS.



CHECK RELATIONSHIP WITH MORE COLUMNS IN HEATMAP.



5

MACHINE LEARNING

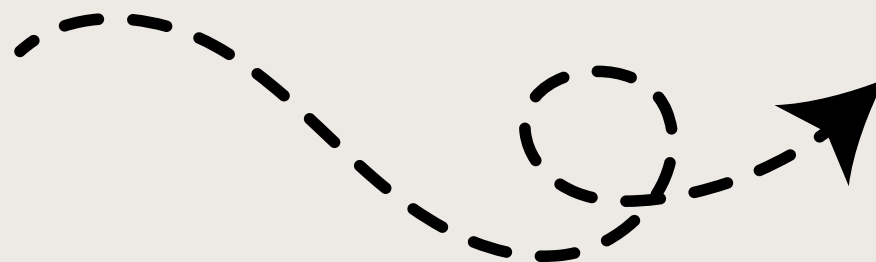


```
7
8 X = df[['view','likes','comments','Flash','more_pro_in_brand','more_pro_in_model','most pro in year','Performances in Likes','likes_views_ratio','comments_views_ratio','comments_likes_ratio','macro_or_micro']]
9 y = df['pro']
10
11 # חלוקת הנתונים לסט האימון וחבדיקה
12 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
13
14 # יצירת ואימון המודל לוגיסטי רגרסיה
15 model = LogisticRegression()
16 model.fit(X_train, y_train)
17
18 # חיזוי התוצאות עבור סט חבדיקה
19 y_pred = model.predict(X_test)
20
21 # חסוואה לערכים של 1
22 y_pred_1 = (y_pred ==1)
23
24 confusion_mtx = confusion_matrix(y_test, y_pred)
25
26 true_negative, false_positive, false_negative, true_positive = confusion_mtx.ravel()
27
28 # Calculate accuracy
29 accuracy = (true_positive + true_negative) / (true_positive + true_negative + false_positive + false_negative)
30
31 # Calculate recall (also known as sensitivity or true positive rate)
32 recall = true_positive / (true_positive + false_negative)
33
34 # Calculate precision (also known as positive predictive value)
35 precision = true_positive / (true_positive + false_positive)
36
37 # F1 score
```

```
5 # קריאת הנתונים מ-csv
```

```
6 df = pd.read_csv("Cleaned_csv_file1.7.csv")
```

```
true_negative 5966
false_positive 1
false_negative 3871
true_positive 2
Accuracy: 0.6065040650406504
Recall: 0.0005163955589981926
Precision: 0.6666666666666666
F-measure: 0.0010319917440660474
```



```
6 df = pd.read_csv("Cleaned_csv_file1.9.2.csv")
```

```
true_negative 4716
false_positive 817
false_negative 2280
true_positive 1474
Accuracy: 0.6665230968019813
Recall: 0.39264784230154504
Precision: 0.6433871671759057
F-measure: 0.4876757650951199
```


5

MACHINE LEARNING



```

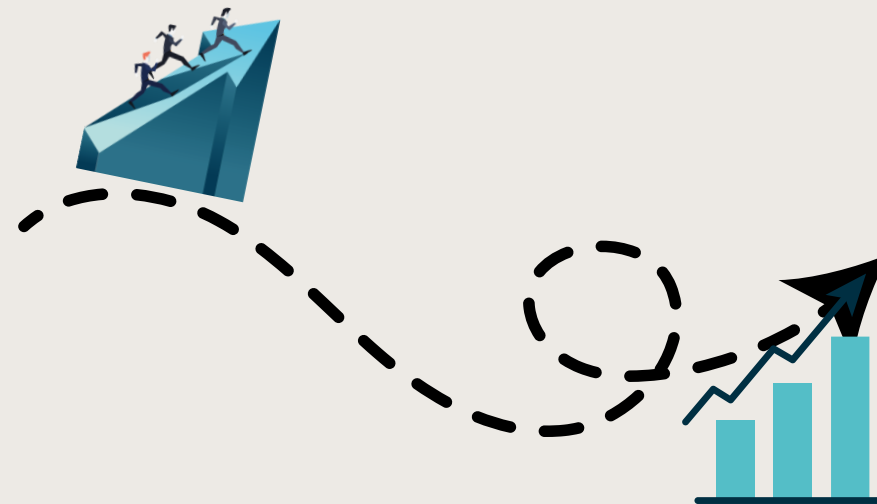
10 X = df[['score_photo_normalized', 'flash', 'macro_or_micro', 'more_pro_in_model', 'most pro in year', 'Performances in Likes', 'likes_views_ratio', 'comments_views_ratio', 'comments_likes_ratio', 'more_pro_in_brand']]
11 y = df['pro']
12
13 scaler = MinMaxScaler()
14 X_scaled = scaler.fit_transform(X)
15
16 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
17
18 model = LogisticRegression()
19 model.fit(X_train, y_train)
20
21 y_pred = model.predict(X_test)
22
23 y_pred_1 = (y_pred == 1)
24
25 confusion_mtx = confusion_matrix(y_test, y_pred)
26
27 true_negative, false_positive, false_negative, true_positive = confusion_mtx.ravel()
28
29 accuracy = (true_positive + true_negative) / (true_positive + true_negative + false_positive + false_negative)
30
31 recall = true_positive / (true_positive + false_negative)
32
33 precision = true_positive / (true_positive + false_positive)
34
35 f_measure = 2 * (precision * recall) / (precision + recall)
36
37 print('true_negative', true_negative)
38 print('false_positive', false_positive)
39 print('false_negative', false_negative)
40 print('true_positive', true_positive)
41 print("Accuracy:", accuracy)
42 print("Recall:", recall)
43 print("Precision:", precision)
44 print("F-measure:", f_measure)

```

```

6 df = pd.read_csv("Cleaned_csv_file1.9.2.csv")
true_negative 4697
false_positive 836
false_negative 2257
true_positive 1497
Accuracy: 0.6669538063960375
Recall: 0.39877464038359084
Precision: 0.6416630947278182
F-measure: 0.4918679152291769

```



```

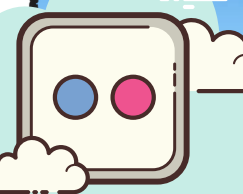
6 df = pd.read_csv("Cleaned_csv_file1.7.csv")
true_negative 5071
false_positive 896
false_negative 2306
true_positive 1567
Accuracy: 0.6745934959349593
Recall: 0.4045959204750839
Precision: 0.6362159967519285
F-measure: 0.4946338383838384

```




Summary

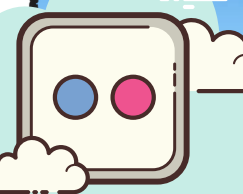
- Flickr is a photo-sharing website for photographers, both amateur and professional.
- It provides a platform for uploading and sharing photos, without distinguishing between free and pro users.
- Users can organize their photos and receive comments from others.
- There is no direct correlation between features like cameras, likes, comments, or views and a user's free or pro status.
- Pro users enjoy additional benefits such as more storage and access to personal statistical information





Summary

- Flickr is a photo-sharing website for photographers, both amateur and professional.
- It provides a platform for uploading and sharing photos, without distinguishing between free and pro users.
- Users can organize their photos and receive comments from others.
- There is no direct correlation between features like cameras, likes, comments, or views and a user's free or pro status.
- Pro users enjoy additional benefits such as more storage and access to personal statistical information



- GeekForGeeks
- Selenium website
- Chat GPT
- Google Bard
- Numpy
- Matplotlib
- StackOverflow
- W3schools
- Wikipedia
- Campus IL
- medium.com
- Flickr |blog

flickr

