The Chinese University of Hong Kong, Shen Zhen

MKT4220 Final Report

BIG DATA MARKETING

Beauty of Pictures: Image Engagement in Cosmetic Industry

Group 1:

Author:	*Student number*:
WeiShiyun	120090564
LiChuyi	120090844
WangHailing	120090308
ZhangJiarui	120030058
LiangTianning	119020280
WuShuo	119020441

Introduction

Referring to the thoughts in the midterm paper, our group try to analyse the image content effects to social media marketing activities. Besides observing image content effects, we especially chose pictures in the cosmetic industry as our research objects. Our data are collected from posts on *Flickr*. Also, we consider the posts' images, text features, and other relevant information together. We then identify parameters for "a best engaging post" and create our model. To ensure the model is more applicable in real life, we test our model on the practically used platform. RED is a social media platform containing user-generated-contents. When users post text and image content about cosmetic products on RED, other users attracted by the recommendations will buy the products. The buying behaviours of users make profits for companies and further earnings for RED. This business cycle on RED motivates many brands to buy placement ads or article ads to promote their cosmetic products. It also justifies our choice to test data on RED.

Since there are many categories in cosmetic products, we study the pictures' characteristics that generate more engagement and explore the difference of effects' characteristics when product types vary. Therefore, our research questions are: What features can cosmetics product pictures create more engagement on social media platforms, and do these affect characteristics vary from different categories of cosmetics products?

Literature Reviews

Prediction of content popularity on social networks helps brand managers to provide better service. In research about popularity prediction of *Flickr* images, the authors point out several effective factors for popular content. Since pictures with their textual contents make the nature of data noisy, they focus on image content along with textual features to design a hybrid deep-based model to address the complexity of data. The result of this study indicates that the image effect interconnects with texts and other components. Because of the complex nature of social media platforms, noise in the dataset requires pre-processing to get an accurate and reasonable model.

Furthermore, in a study about advertisement recommendation, the authors present a new model named IPARS (image-based personalized advertisement recommendation system) to identify target customers on social media. They use image processing and machine learning techniques for online advertisement. Among the model, there are two layers. The shared keywords among both layers link the layers and create the relationship between images and users to find potential customers. According to this research, researchers are finding connected elements in two layers are important when searching for the best audience for an ad. This research further proves the business value of a multi-layer system when matching textual information in the image with customers' sharing habits.

Since there are pictures in different fields on *Flickr*, the detection in one specific area would make our study more concrete. Extracting image features in large social media datasets is difficult; therefore, many researchers are devoted to filtering information before constructing models. For example, in one study about identifying Europe's protected areas, the authors developed a robust processing pipeline to extract image features from the posts.

Based on these papers, our study will study image features based on image features (image quality, colorfulness, and face detection) and post information (tag numbers, commercial permission, and published date). Our contribution is identifying the image, text features, and other relevant details of the posts to find parameters for the best engaging model and test the model on the practically used platform RED to provide valuable suggestions for beauty brands when they run ads.

Model

1. Model selection

"These measures are discrete and non-negative, a count model (e.g., Poisson regression, is more appropriate than a linear regression model."

"we have to extend the standard univariate count regression model to account for ……: (1) the presence of excess zeros in our data."

From our midterm essay, we choose count model instead of linear regression model to fit our discrete measurements: comments, favorites and views for Flickr dataset, and likes for RED dataset. Since we have guaranteed that number of views is larger than zero during the process of data crawling, we directly choose Poisson regression model to conduct our analysis.

2. Model structure

2.1 Model Illustration

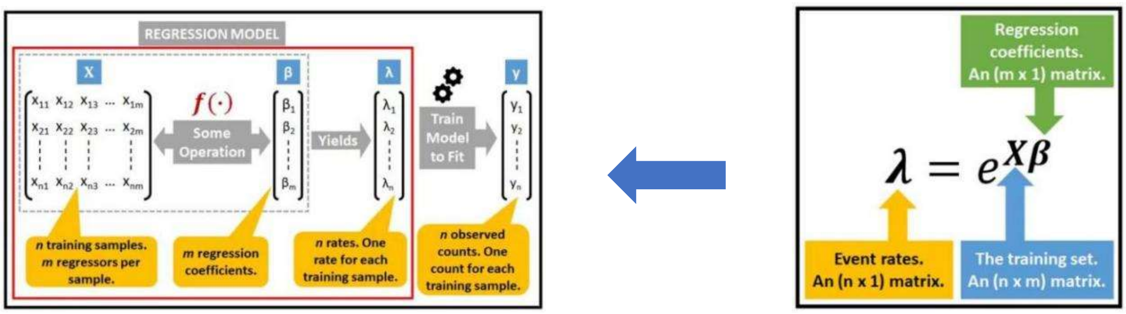


Figure 1: Poisson regression model

Suppose we have n training samples with m features as input X, which is a n*m matrix, we take its exponential result after multiplying β to get the event rate λ. After that, we fit those data into train model in order to get our Poisson regression model.

The formula is as follows: $\lambda_i = \exp(\beta_0 + \beta_1 x_i + \cdots + \beta_m x_m)$

$$P(Y_i = y_i) = \exp(-\lambda_i)\frac{\lambda_i^{y_i}}{y_i}$$

Parameters β_0, β_1, … β_m are those we need to fit in our model.

2.2 Tool selection

We use *statsmodels* in python library as our statistical model tool and implement its Poisson regression function to get our result in a summarized table.

Data

1. Data source and selection

Since our research requires specific data, we choose to crawl data from the internet.

1.1 Training Data

We collect data from *Flickr.com* as our training data set. This website allows searching pictures with keywords when doing data crawling. It also provides an official API for data crawling. This ensures the size and quantity of our training dataset, which improves the accuracy of our machine-learning model.

According to the information displayed on the website, we decide to use the number of comments, views, and favorites as the measurement for social-engagement of users.

1.2 Testing Data

We also collect some data from RED as our testing data set. RED is a widely-used social and promotion platform in China. So, it's worthy to study what kind of pictures are more attractive to users on it. Moreover, its main interface is designed to be the "Picture Waterfall flow" style, which makes the App well-fit with our research.

Considering that RED do not support crawling of large scale of data, we use it as the testing data for our model. And for available reasons, we choose the number of Likes as the measurement for social-engagement.

2. Data crawling

2.1 Crawling Methods

Training data: We directly use the official API of *Flickr*. We tuned parameters, "tags" and "text", to grab pictures according to our requirements. Cosmetic category is set as "tags", including "lipsticks", "foundation", and "makeup". And for "text", 4 cosmetics brands with the same consumption level of main buyers are selected. They are *Lancome*, *Estee Lauder*, *Armani*, and *L'Oréal*.

Testing data: We search posts by phrases combined with brand and category by special app.

And then, we decode the data and extract the needed information using Python packages: *BeautifulSoup* and *etree*. Finally, we saved the information of figures in csv files and saved the figures in jpg format.

2.2 Crawling Results

Training data: We captured the basic information of pictures and the text of their posts. The basic information of pictures is their *post title, id, id of the uploader, size (height and width), URL, upload date,* and *commercial license*. The text of picture-posts mainly includes *tags* and *description*.

Testing data: Similarly, we grab the basic information of pictures and their posts text from RED.

The basic information is the *upload date* and *user id of the uploader* of the pictures. And the text of post is the full content uploaded on the platform.

3. Data pre-processing

3.1 Filtering

We filtered those redundant images with high similarity to others and irrelevant images which do not match our searching tags and texts manually.

Before filtering, we have 2679 images & information in total.

Table 1: data before filtering

	Lancôme	Estee Lauder	Armani	L'Oréal	All
Lipstick	124	160	22	256	562
Foundation	21	121	8	107	257
Make-up	888	350	622	-	1860

After filtering, we have 820 images & information in total. We merge them into three csv files: makeup_all, lipstick_all, and foundation_all according to the categories.

3.2 Image processing

We implement the open-source Python library *OpenCV* to conduct image processing for our images. It adopts machine learning algorithms to process images and is very popular in computer vision field. We mainly analyze our images in three aspects.

- Human face detection:

We use Haar-cascade Classifier to test whether the image has human face or not. It is a machine learning algorithm where we train a cascade function with a large number of images. There are different types of cascade classifiers depending on the target object, here we will use a classifier that considers faces to identify them as the target object. In order to make the results clearer, we record the number of detected faces as 1 if it is greater than or equal to 1, otherwise it is recorded as 0.

- Quality (Sharpness):

We use Laplacian gradient function to test the sharpness of an image.

$$z(x,y) = g(x-1,y) + g(x+1,y) + g(x,y-1) + g(x,y+1) - 4g(x,y)$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Figure 2: Laplacian operator

The high frequency components of the image can be obtained by template convolution of the image using Laplacian operator, and then the high frequency components of the image can be summed up. The sum of the high frequency components is used as the image definition evaluation standard.

For one m×n pixel image, the brightness of each pixel is g(x,y), and the value of each pixel after filtering template convolution is z(x,y).

Image definition evaluation function: $f = \frac{1}{MN}\sum_{(x,y)} z(x,y)$

- Color richness:

$$rg = R - G$$

$$yb = \frac{1}{2}(R+G) - B$$

$$\sigma_{rgyb} = \sqrt{\sigma_{rg}^2 + \sigma_{yb}^2}$$

$$\mu_{rgyb} = \sqrt{\mu_{rg}^2 + \mu_{yb}^2}$$

$$C = \sigma_{rgyb} + 0.3 * \mu_{rgyb}$$

We use Hasler and Süsstrunk's study result to get the color richness of images. They classified color richness into 7 levels and asked 20 people to rate 84 images on a scale of 1 to 7. The final analysis of this survey data revealed the following formula for calculating the colorfulness of the images. And the final C is the indicator variable for the color richness of the images (where sigma and miu represent the standard deviation and mean respectively).

3.3 Text processing

Description: We classify data into two types and label them respectively, 0 for those without description, and 1 for those with descriptions.

- Remark:

We tried to extract keywords in description contents by machine learning techniques and successfully get them in column "keywords". However, we manually check that those keywords do not correlate with our purpose of examining consumers' attitude towards cosmetics well. So, we finally decided not to use it, we just use the existence of descriptions or not as one of the features.

- Sample of keywords list:

```
["Estée Lauder's", '1 July', '1908-2004', 'Estee Lauder', 'John Schotz', 'two', 'Estée Lauder', '1946', 'Joseph', 'A year later', 'Saks Fifth Avenue', '800', 'two days', 'The Lauder Foundation', 'the 1960s', '7 decades', 'the present day', '0.2mm']
```

Figure 3: sample of one extracted keywords list

Upload time: We sort the uploaded time of all posts by timestamp value and record in date_tuple column by sorted order. The data frame view of posts after text processing is as follows:

tags	tags_percent	...	description	date	favorites	NonCommercial	quality	keywords	colorful	face	description_exist	date_tuple
3	66.666667	...	Finding a foundation that flatters, brightens,...	2007-01-01 00:00:00	0	1	204	['Achilles', 'Rimmel, Mac', 'Max Factor', 'Est...	33.282292	0	1	0
16	12.500000	...	estee lauder eye shaddow : new pallet : car b...	2007-10-07 18:26:00	0	1	1492	['estee lauder eye', '50p', 'german', 'un', 'a...	112.072172	1	1	1
17	11.764706	...	estee lauder eye shaddow and a german make fou...	2007-10-07 18:26:00	0	1	535	['estee lauder eye', 'german', 'un']	89.192961	1	1	2
12	16.666667	...	? L'oreal Colour Riche in 115- Laetitia's Champ...	2008-07-29 10:45:00	2	1	1004	['115', 'Champagne', 'Rimmel', '010', 'True Ma...	68.707931	1	1	3
5	40.000000	...	eeeeeeeeeee chegou hoje minha base! (dancinha ...	2008-09-03 01:20:00	2	1	60	['Agora', 'soh ouvi', 'respeito', 'vcs', 'ache...	12.297644	0	1	4

Figure 4: sample of upload date timestamp and its sorted order

3.4 Model fitting

Sample code and output:

```
poisson = sm.Poisson(Y, X).fit()
print(poisson.summary())
```

```
Optimization terminated successfully.
         Current function value: 449.364307
         Iterations 10
                          Poisson Regression Results
==============================================================================
Dep. Variable:                  views   No. Observations:                  106
Model:                        Poisson   Df Residuals:                       99
Method:                           MLE   Df Model:                            6
Date:                Mon, 14 Nov 2022   Pseudo R-squ.:                  0.6681
Time:                        11:07:32   Log-Likelihood:                -47633.
converged:                       True   LL-Null:                   -1.4353e+05
Covariance Type:            nonrobust   LLR p-value:                     0.000
===================================================================================
                      coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------
tags                0.0521      0.000    243.201      0.000       0.052       0.053
NonCommercial       5.6396      0.012    454.613      0.000       5.615       5.664
quality             0.0002   2.24e-06     71.535      0.000       0.000       0.000
colorful           -0.0138      0.000    -91.932      0.000      -0.014      -0.013
face               -0.1875      0.006    -33.095      0.000      -0.199      -0.176
description_exist   2.4923      0.009    286.492      0.000       2.475       2.509
date_tuple         -0.0222   9.88e-05   -224.345      0.000      -0.022      -0.022
===================================================================================
```

Figure 5: sample output coefficients and statistics

The pseudo-R square can serve as a statistic for us to check the accuracy of our model. The coefficients provided in the table represents the positive correlation/negative correlation between each feature and the respective measurement of user engagement by their signs.

Results & Conclusion

1. General result for makeup: (Answering research question 1)

Here is the Poisson regression result from makeup_all.csv which consists of all cosmetic posts.

Table 2: general results for three measurements and six features

make-up	tags	quality	colorful	face	description_exist	date_tuple
favorites	0.0163	-9.635e-05	0.0111	-0.2260	0.5476	-0.0014
views	-0.0004	4.514e-06	0.0108	-0.0929	1.1003	-0.0049
comments	0.0019	-0.0005	0.0126	0.0863	0.6174	-0.0042

Positive-correlated:

Those results correspond with our prediction that texts and tags (#) can have positive effect on user engagement for social media posts. Also, in terms of images, more colorful images will receive higher engagement level on social media platform.

Negative-correlated:

Regarding upload date, latest uploaded images receive lower measurements. This result also matches our expectation, since historical engagement data can be accumulated. The longer a post is released, the higher the cumulative engagement will be.

Others:

High image quality will have negative effect on favorites and comments but positive effect on views. Compared with the results in our mid-presentation paper, we found the different negative effect on users' favorites and comments. Despite being viewed by more users, those images do not turn such views into actual interaction. This makes us reflect on the advertising effect of images.

Having human face will have negative effect on favorites and views but positive effect on comments. Since *Flickr.com* serves as an image sharing website with social functions, showing one's human face will inspire user interaction, which is common on social media. However, negative effect on favorites and views may be due to the personal information included in images with human face. Others may try to avoid engagement with those images out of consideration of personal privacy.

2. Specified result regarding different categories of cosmetics products:

(Answering research question 2)

Here is the Poisson regression result from foundation_all.csv and lipstick_all.csv which consists of all posts with *#foundation* and *# lipstick* respectively.

Table 3: specific result for foundation and lipstick

foundation	tags	quality	colorful	face	description_exist	date_tuple
favorite	0.0590	0.0005	-0.0320	-0.4072	2.1359	-0.0670
views	0.0521	0.0002	-0.0038	-0.1875	2.4923	-0.0222
comments	0.0391	0.0006	-0.0512	-1.4818	3.8128	-0.0809
lipstick	**tags**	**quality**	**colorful**	**face**	**description_exist**	**date_tuple**
favorite	0.0198	6.833e-05	-0.0009	0.4490	0.7964	-0.0015
views	0.0275	1.23e-05	0.0126	0.3775	1.0540	-0.0030
comments	0.0019	-0.0005	0.0126	0.0863	0.6174	-0.0042

We try to focus on image effect in terms of those two categories of cosmetics products.

- Human face——negative for foundation, positive for lipstick

Surprisingly, we found that when sharing lipsticks, presence of human face can have positive effect on user engagement. However, for those sharing foundations, having human face will have negative effects. This inspires us that when advertising cosmetic products focusing on different parts on human face, we may need to adopt various strategies in order to increase user engagement.

- Image quality——generally positive for both, but negative in comments for lipstick

Generally, image quality still shows positive effect on user engagement. This fits well our expectation.

- Colorful images——negative for foundation, generally positive for lipstick while showing a negative effect on favorites

This may correspond with the characteristics of lipstick and foundation. Since lipstick often has a bright color such as red and pink, high saturation in images may make lipstick more attractive. As for foundation, they serve as the bottom makeup in cosmetics and has the color close to skin tone, so colorful images may mislead users from their actual effect.

3. Testing dataset result

In order to test whether our prediction of image features' effects on user engagement above is applicable on other platforms, we use coefficients trained from *Flickr.com* to predict the measurement of likes in test dataset - RED dataset.

Sample of test images and information is as follows:

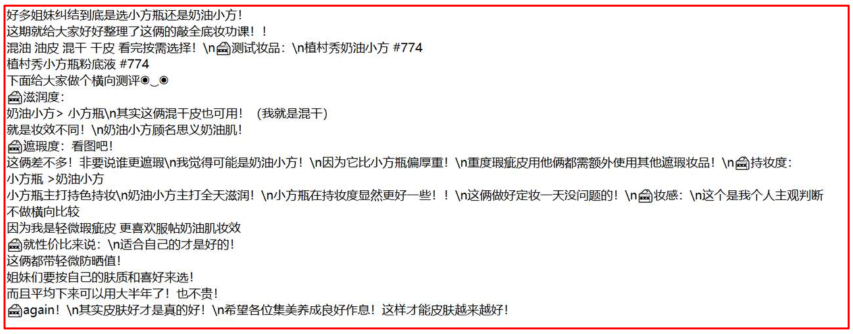

好多姐妹纠结到底是选小方瓶还是奶油小方！
这期就给大家好好整理了这俩的敲全底妆功课！！
混油 油皮 混干 干皮 看完按需选择！\n测试妆品：\n植村秀奶油小方 #774
植村秀小方瓶粉底液 #774
下面给大家做个横向测评◉‿◉
滋润度：
奶油小方> 小方瓶\n其实这俩混干皮也可用！（我就是混干）
就是妆效不同！\n奶油小方顾名思义奶油肌！
遮瑕度：看图吧！
这俩差不多！非要说谁更遮瑕\n我觉得可能是奶油小方！\n因为它比小方瓶偏厚重！\n重度瑕疵皮用他俩都需额外使用其他遮瑕妆品！\n持妆度：
小方瓶 >奶油小方
小方瓶主打持色持妆\n奶油小方主打全天滋润！\n小方瓶在持妆度显然更好一些！！\n这俩做好定妆一天没问题的！\n妆感：\n这个是我个人主观判断
不做横向比较
因为我是轻微瑕疵皮 更喜欢服帖奶油肌妆效
就性价比来说：\n适合自己的才是好的！
这俩都带轻微防晒值！
姐妹们要按自己的肤质和喜好来选！
而且平均下来可以用大半年了！也不贵！
again！\n其实皮肤好才是真的好！\n希望各位集美养成良好作息！这样才能皮肤越来越好！

Figure 6: view of RED images Figure 7: view of RED posts

Similarly, we merge data into two excel files by categories: test_lipstick1 and test_foundation1.

For lipstick:

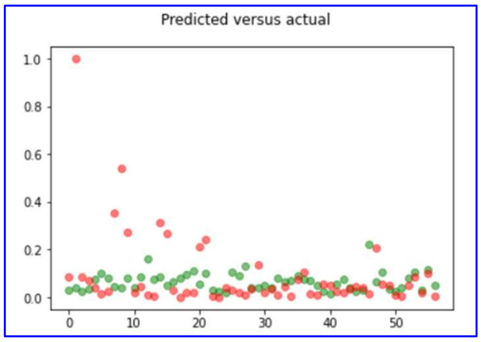


Figure 8: After data normalization

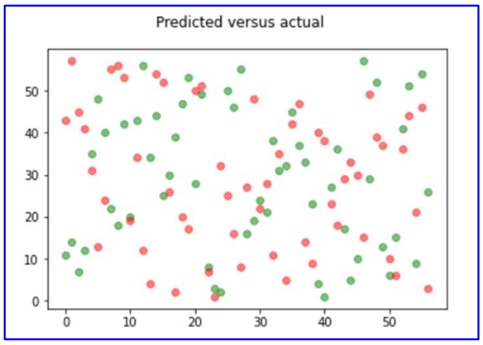


Figure 9: After ranked

The red points are actual counts from RED, and the green points are predicted counts of RED from *Flickr* data set. Due to the difference of cardinality for two data sets (RED with average likes more than 10,000 and *Flickr* with average engagements less than 100), we first normalize those engagement data Y before fitting.

However, the predicted result in the left picture still shows a significant difference. We tried to rank the engagements and plot the ranked data on the right figure, but the effect of fitness is still low. Finally, we conducted rank test and the p-value is 0.010677998035961233, which is very small. This means that we need to reject zero assumption and accept the null assumption that our model in training data set is not applicable to test data set in terms of lipstick categories.

For foundation:

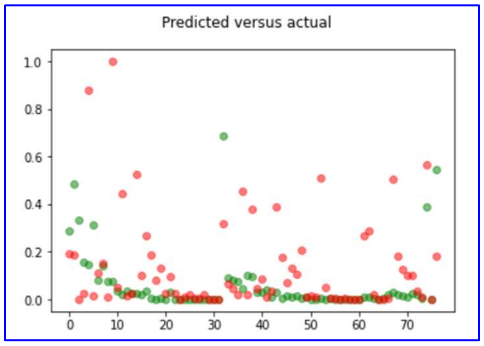


Figure 10: After data normalization

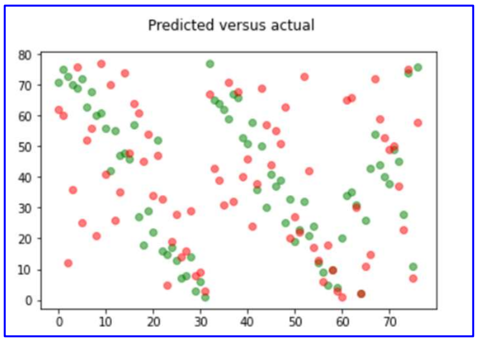


Figure 11: After ranked

We have similar problems with our data as above. Adopting the same strategies of data normalization and rank sum test, we have p-value=0.0030633931339710003. We can see that p value is also very small, so our model in training data set is not applicable to test data set in terms of foundation categories.

This inapplicability may be owing to culture difference. Flickr is a Japanese website widely used by foreigners, while RED is only popular in China. Therefore, people with different culture background may prefer to different kinds of pictures.

4. Conclusion

1. In cosmetics industry, image characteristics can have effect on our measurement of user engagement, while text characteristics still play roles on user engagement on social media.

Specifically, colorfulness has positive effect on all three kinds of user engagement. Image quality has negative effect on favorites and comments but positive effect on views. Existence of human face has negative effect on favorites and views but positive effect on comments.

2. For two categories of make ups: foundation and lipstick, image features' effect on user engagement varies. Despite image quality having positive effect on both foundation and makeup posts' engagement, existence of human face shows negative effect for foundation but positive effect for lipstick. Also, colorfulness of images appears negative for foundation, but generally positive for lipstick except showing a negative effect on favorites.
3. Image characteristic effects on user engagement differ in different social media platforms.

5. Limitation

- Image-text Fitness is not enough

We have not tested the relevance between image and text of the posts, sometimes users would upload the texts irrelevant with image and makes prediction inaccurate.

- The test data on RED is not sufficient

Since RED is a more practical platform for brands, we need to collect more data.

6. Contribution

In summary, we collect our posts from *Flickr.com* as training set and RED application as test set. Then we filter some irrelevant and use machine learning tools to conduct image processing for the images in those posts to capture the sharpness, color richness and detect human face. After that, we do some data cleaning to fit our factors into the model. For model selection, we select Poisson regression in count model to do our analysis.

The result can be divided into two parts. Regarding research question 1, we found that image characteristics do have effect on our measurement of engagement. And regrading research question 2, for two types of makeups: foundation and lipstick, they have different engagement effects on some image features. For example, human face shows negative effect on user engagement for foundation, but show positive effect for lipstick. This provides a view for future makeup brands in terms of advertising on social media, since they may adopt different strategies to select and post images in terms of different types of cosmetics.

References

Alijani, S., Tanha, J., & Mohammadkhnli, L. (2022). An Ensemble of Deep Learning Algorithms for Popularity Prediction of Flickr Images. Data Science on Multimedia Data: Challenges and Applications.

C. Hartmann, M., Koblet, O., F. baer, M., & S. Purves, R. (2022). Automated Motif Identification: Analyzing Flickr Images to Identify Popular Viewpoints in Europe's Protected Areas. ScienceDirect.

Jouyandeh, F., & Zadeh, P. M. (2022). IPARS: An Image-Based Personalized Advertisement Recommendation System on Social Network. ScienceDirect.

Appendix: code (pictures or module)

Appendix I: code for data crawling

get picture list

In [1]:
```
import flickrapi
import urllib.request
import os
import sys
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
def get_pictures(searchTag,searchText):
    # connect Flickr API
    API_KEY = "b7fe524cecd902e07dc95c8ddb1d8bb3"
    API_SECRET = "7f82d0d5a0c79e95"
    flickr = flickrapi.FlickrAPI(API_KEY, API_SECRET, cache=True)

    '''
    Parameters of flickrapi.FlickrAPI.walk():
    # tags
    # text
    # sort = [date-posted-asc, date-posted-desc, date-taken-asc, date-taken-desc, interestingness-desc, interestingness-asc, relevance]
    # media = [photos, videos]
    # per_page: Number of photos to return per page. It defaults to 100. The maximum allowed value is 500.
    # page: The page of results to return. If this argument is omitted, it defaults to 1
    # extras = [description, license, date_upload, date_taken, owner_name, icon_server, original_format, last_update,
               geo, tags, machine_tags, o_dims, views, media, path_alias, url_sq, url_t, url_s, url_q, url_m, url_n,
               url_z, url_c, url_l, url_o]
    '''
    # search for pictures
    try:
        photos = flickr.walk(tags=searchTag,text=searchText, sort='date-posted-desc', media='photos', per_page=200, pages=5, extras='url_c')
    except Exception as e:
        print('Error')

    data_columns = ['title','photo_id','secret','owner','photo_height','photo_width','url_photo']
    df = pd.DataFrame(columns=data_columns, index=[])

    for photo in photos:
        # get basic info of photos
        title= photo.get('title')
        id_pho = photo.get('id')
        owner = photo.get('owner')
        server = photo.get('server')
        secret = photo.get('secret')
```

```
    for photo in photos:
        # get basic info of photos
        title= photo.get('title')
        id_pho = photo.get('id')
        owner = photo.get('owner')
        server = photo.get('server')
        secret = photo.get('secret')
        farm = photo.get('farm')
        photo_height = photo.get('height_c')
        photo_width = photo.get('width_c')
        # get url of photo
        url=photo.get('url_c')

        if(str(url) == "None"):
            print("It's None!")
        else:
            data = pd.Series({'title':title, 'photo_id':id_pho, 'secret':secret, 'owner':owner, 'photo_height':photo_height, 'photo_width':photo_width, 'url_photo':url})
            df = df.append(data, ignore_index=True)
            print(url)

    return df
```

get more information

In [3]:
```
## get more information of these photos (FlickrAPI)

import time

def get_moreInfo(df):
    # connect Flickr API
    API_KEY = "b7fe524cecd902e07dc95c8ddb1d8bb3"
    API_SECRET = "7f82d0d5a0c79e95"
    flickr = flickrapi.FlickrAPI(API_KEY, API_SECRET, cache=True)

    pho_id = df['photo_id']
    pho_secret = df['secret']
    url_L = df['url_photo']
    path = "D:\Selina\LGU\MKT4220\FINAL Project\Flickr_data\ "

    data_columns = ['general_info','favorites','licenses']
    info_df = pd.DataFrame(columns=data_columns, index=range(len(df)))

    for i in range(len(df)):
        Id = pho_id[i]
        Secret = pho_secret[i]
        try:

            info = flickr.photos.getInfo(photo_id=Id, secret=Secret)
            info2 = flickr.photos.getFavorites(photo_id=Id, secret=Secret)
            info3 = flickr.photos.licenses.getInfo(photo_id=Id, secret=Secret)

            # download photos in the local file
            url = url_L[i]
            count = i
            urllib.request.urlretrieve(url, path+tag+text+str(count).zfill(7) + "." + os.path.basename(url).split(".")[1])

            # store information in dataframe
            info_df.loc[i,'general_info'] = info
            info_df.loc[i,'favorites'] = info2
            info_df.loc[i,'licenses'] = info3
            print('Done'+str(i), end=' ')

        except:
            info_df.loc[i,'general_info'] = None
            info_df.loc[i,'favorites'] = None
            info_df.loc[i,'licenses'] = None
            print('error')

    print('\n',"Done all!")
    return info_df
```

preprocessing the features

In [4]:
```
## Use Beautiful Soup to process the features and store in dataframe

from bs4 import BeautifulSoup
from lxml import etree
import numpy as np

def store(df, info_df):
    for i in range(len(df)):
        info1 = info_df.iloc[i,0]
        info2 = info_df.iloc[i,1]
        info3 = info_df.iloc[i,2]

        # preprocessing
        if info1 != None:
            info = etree.tostring(info1,encoding="utf-8").decode()
            soup = BeautifulSoup(info,'lxml')
            views_n = soup.photo['views']
            comments_n = soup.photo.comments.string
            description = soup.photo.description.string
            taken_date = soup.dates['taken']
            tags = soup.find_all('tag')
            url_post = soup.photo.urls.url.string
            # store in dataframe
            df.loc[i,'url_post'] = url_post
            df.loc[i,'tags'] = int(len(tags))
            if len(tags) != 0:
                df.loc[i,'tags_percent'] = (2/len(tags))*100
            elif len(tags) == 0:
                df.loc[i,'tags_percent'] = 0
```

```
            elif len(tags) == 0:
                df.loc[i,'tags_percent'] = 0
            df.loc[i,'views'] = views_n
            df.loc[i,'comments'] = comments_n
            df.loc[i,'description'] = description
            df.loc[i,'date'] = taken_date
        else:
            df.loc[i,'url_post'] = None
            df.loc[i,'tags'] = None
            df.loc[i,'tags_percent'] = None
            df.loc[i,'views'] = None
            df.loc[i,'comments'] = None
            df.loc[i,'description'] = None
            df.loc[i,'date'] = None

        if info2 != None:
            info2 = etree.tostring(info2,encoding="utf-8").decode()
            soup2 = BeautifulSoup(info2,'lxml')
            favorites_n = soup2.photo['total']
            # store in dataframe
            df.loc[i,'favorites'] = favorites_n

        if info3 != None:
            info3 = etree.tostring(info3,encoding="utf-8").decode()
            soup3 = BeautifulSoup(info3,'lxml')
            licenses = soup3.licenses.find_all('license')
            licenses_L = []
            for m in range(len(licenses)):
                licenses_L.append(licenses[m]['id'])
            if '1' in licenses_L or '2' in licenses_L or '3' in licenses_L:
                NonCommercial = '1'
            else:
                NonCommercial = '0'
            # store in dataframe
            df.loc[i,'NonCommercial'] = NonCommercial

        print('Done'+str(i), end=' ')

    return df
```

Run

In [5]:

```
Tag_L = ['lipstick', 'foundation','makeup', '']
Text_L = ['canmake','lancome','estee lauder','armani',"L'Oreal",
          'canmake foundation','canmake makeup']

tag = Tag_L[0] # generally: 0-2
text = Text_L[1] # generally: 0-4
df_pictures = get_pictures(searchTag=tag, searchText=text)

# test howmany pictures are found
df_pictures
```

In [6]:

```
filename = 'pictureList_'+tag+"_"+text+".csv"
df_pictures.to_csv(filename,index=None)
df_pictures
```

In [7]:
```
df_info = get_moreInfo(df_pictures)
filename2 = 'pictureInfo_'+tag+"_"+text+".csv"
df_info.to_csv(filename2,index=None)
```

In [8]:
```
# databricks
dfInfo = pd.read_csv(filename2)
df_final = store(df_pictures, df_info)
filename3 = 'Pictures_'+tag+"_"+text+".csv"
df_final.to_csv(filename3,index=None)
```

Appendix II: data pre-processing

```
import os
from PIL import Image
import cv2

def traverse_dir_files(root_dir, ext=None):
    paths_list = []
    full_list = []
    for parent, _, fileNames in os.walk(root_dir):
        for name in fileNames:
            if name.startswith('.') :
                continue
            if ext:
                if name.endswith(tuple(ext)):
                    paths_list.append(parent)
                    full_list.append(os.path.join(parent, name))
            else:
                paths_list.append(parent)
                full_list.append(os.path.join(parent, name))
    return paths_list,full_list

p_list,f_list = traverse_dir_files('G:/MKT/train code/Flick','.jpg')
```

```
def get_picture_sharpness(filestr):
    image = cv2.imread(filestr)
    image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    image_var = str(int(cv2.Laplacian(image_gray, cv2.CV_64F).var()))

    font_face = cv2.FONT_HERSHEY_COMPLEX
    font_scale = 1
    thickness = 1
    baseline = 0

    var_size = cv2.getTextSize(image_var, font_face, font_scale, thickness)
    draw_start_point = (20, var_size[0][1] + 10)
    cv2.putText(image, image_var, draw_start_point, font_face, font_scale, (0,0,255), thickness)

    cv2.waitKey(0)

    return image_var

txt = open("G:/MKT/train code/quality.txt", 'w')
for i in range(len(f_list)):
    file_path = f_list.pop(0)
    filestr = file_path.replace('\\', '/')
    print(filestr)
    txt.write(file_path+","+get_picture_sharpness(filestr) + '\n')
txt.close()
```

```
import cv2
import numpy as np

def image_colorfulness(image):
    (B, G, R) = cv2.split(image.astype("float"))

    #rg = R - G
    rg = np.absolute(R - G)

    #yb = 0.5 * (R + G) - B
    yb = np.absolute(0.5 * (R + G) - B)

    (rbMean, rbStd) = (np.mean(rg), np.std(rg))
    (ybMean, ybStd) = (np.mean(yb), np.std(yb))

    stdRoot = np.sqrt((rbStd ** 2) + (ybStd ** 2))
    meanRoot = np.sqrt((rbMean ** 2) + (ybMean ** 2))

    return stdRoot + (0.3 * meanRoot)
color = []
for i in range(len(f_list)):
    image = cv2.imread(f_list[i])
    color.append(image_colorfulness(image))
print(color)
```

```
import cv2
import matplotlib.pyplot as plt

face_cascade = cv2.CascadeClassifier(r'C:/Users/[redacted]
l = []
for i in range(len(f_list)):
    img = cv2.imread(f_list[i])
    faces = face_cascade.detectMultiScale(image = img, scaleFactor = 1.1, minNeighbors = 5)
        # face numbers
    if(len(faces) < 1):
        l.append(0)
    else:
        l.append(1)
print(l)
```

deal with description

In [5]:
```
df.insert(loc=20, column='description_exist', value=1)
df["description"].fillna(0, inplace = True)
for i in range(449):
    if df["description"][i] == 0:
        df.loc[i,"description_exist"] = 0
```

In [6]:
```
df1.insert(loc=20, column='description_exist', value=1)
df1["description"].fillna(0, inplace = True)
for i in range(265):
    if df1["description"][i] == 0:
        df1.loc[i,"description_exist"] = 0
```

In [7]:
```
df2.insert(loc=20, column='description_exist', value=1)
df2["description"].fillna(0, inplace = True)
for i in range(106):
    if df2["description"][i] == 0:
        df2.loc[i,"description_exist"] = 0
```

deal with timestamp

In [8]:
```
import time
```

In [9]:
```
df.sort_values(by='date',axis=0,ascending=True, inplace=True)
date_list = [i for i in range(449)]
print(date_list)
df.insert(loc=21, column='date_tuple', value=date_list)
df.head()
```

...

In [10]:
```
df1.sort_values(by='date',axis=0,ascending=True, inplace=True)
date_list = [i for i in range(265)]
print(date_list)
df1.insert(loc=21, column='date_tuple', value=date_list)
df1.head()
```

...

In [11]:
```
df2.sort_values(by='date',axis=0,ascending=True, inplace=True)
date_list = [i for i in range(106)]
df2.insert(loc=21, column='date_tuple', value=date_list)
df2.head()
```

Appendix III: code for model fitting and testing

- **Model fitting**

Makeup

In [13]:
```
X = df[['tags','quality','colorful','face',"description_exist","date_tuple"]]
Y = df['favorites']
poisson = sm.Poisson(Y, X).fit()
print(poisson.summary())
```

```
Optimization terminated successfully.
         Current function value: 4.187690
         Iterations 7
                          Poisson Regression Results
==============================================================================
Dep. Variable:              favorites   No. Observations:                  449
Model:                        Poisson   Df Residuals:                      443
Method:                           MLE   Df Model:                            5
Date:                Sun, 13 Nov 2022   Pseudo R-squ.:                 0.04912
Time:                        23:54:08   Log-Likelihood:                -1880.3
converged:                       True   LL-Null:                       -1977.4
Covariance Type:            nonrobust   LLR p-value:                 4.751e-40
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
tags                  0.0163      0.002      7.267      0.000       0.012       0.021
quality           -9.635e-05   2.73e-05     -3.528      0.000      -0.000   -4.28e-05
colorful              0.0111      0.001      9.393      0.000       0.009       0.013
face                 -0.2260      0.066     -3.411      0.001      -0.356      -0.096
description_exist     0.5476      0.081      6.767      0.000       0.389       0.706
date_tuple           -0.0014      0.000     -5.768      0.000      -0.002      -0.001
=====================================================================================
```

In [15]:
```
X = df[['tags','quality','colorful','face',"description_exist","date_tuple"]]
Y = df['views']
poisson = sm.Poisson(Y, X).fit()
print(poisson.summary())
```

```
Optimization terminated successfully.
         Current function value: 1381.619052
         Iterations 32
                          Poisson Regression Results
==============================================================================
Dep. Variable:                  views   No. Observations:                  449
Model:                        Poisson   Df Residuals:                      442
Method:                           MLE   Df Model:                            6
Date:                Mon, 14 Nov 2022   Pseudo R-squ.:                  0.2052
Time:                        11:05:16   Log-Likelihood:            -6.2035e+05
converged:                       True   LL-Null:                   -7.8055e+05
Covariance Type:            nonrobust   LLR p-value:                     0.000
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
tags                 -0.0004   9.06e-05     -4.476      0.000      -0.001      -0.000
NonCommercial         6.7092      0.006   1186.099      0.000       6.698       6.720
quality            4.514e-06   8.18e-07      5.517      0.000    2.91e-06    6.12e-06
colorful              0.0108   5.06e-05    214.083      0.000       0.011       0.011
face                 -0.0929      0.003    -34.818      0.000      -0.098      -0.088
description_exist     1.1003      0.005    232.092      0.000       1.091       1.110
date_tuple           -0.0049   1.19e-05   -415.829      0.000      -0.005      -0.005
=====================================================================================
```

In [16]:
```
X = df[['tags','quality','colorful','face',"description_exist","date_tuple"]]
Y = df['comments']
poisson = sm.Poisson(Y, X).fit()
print(poisson.summary())
```

```
Optimization terminated successfully.
         Current function value: 2.124472
         Iterations 7
                          Poisson Regression Results
==============================================================================
Dep. Variable:               comments   No. Observations:                  449
Model:                        Poisson   Df Residuals:                      443
Method:                           MLE   Df Model:                            5
Date:                Mon, 14 Nov 2022   Pseudo R-squ.:                  0.1096
Time:                        11:05:33   Log-Likelihood:                -953.89
converged:                       True   LL-Null:                       -1071.3
Covariance Type:            nonrobust   LLR p-value:                 9.947e-49
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
tags                  0.0019      0.004      0.501      0.617      -0.006       0.009
quality              -0.0005   6.72e-05     -6.870      0.000      -0.001      -0.000
colorful              0.0126      0.002      7.559      0.000       0.009       0.016
face                  0.0863      0.097      0.887      0.375      -0.104       0.277
description_exist     0.6174      0.114      5.423      0.000       0.394       0.841
date_tuple           -0.0042      0.000    -10.683      0.000      -0.005      -0.003
=====================================================================================
```

Lipstick

In [14]:
```
X = df1[['tags','quality','colorful','face',"description_exist","date_tuple"]]
Y = df1['favorites']
poisson = sm.Poisson(Y, X).fit()
print(poisson.summary())
```

```
Optimization terminated successfully.
         Current function value: 6.138733
         Iterations 11
                          Poisson Regression Results
==============================================================================
Dep. Variable:              favorites   No. Observations:                  265
Model:                        Poisson   Df Residuals:                      259
Method:                           MLE   Df Model:                            5
Date:                Sun, 13 Nov 2022   Pseudo R-squ.:                 0.03946
Time:                        23:54:11   Log-Likelihood:                -1626.8
converged:                       True   LL-Null:                       -1693.6
Covariance Type:            nonrobust   LLR p-value:                 3.948e-27
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
tags                  0.0198      0.002      8.876      0.000       0.015       0.024
quality            6.833e-05   2.03e-05      3.365      0.001    2.85e-05       0.000
colorful             -0.0009      0.002     -0.560      0.575      -0.004       0.002
face                  0.4490      0.071      6.296      0.000       0.309       0.589
description_exist     0.7964      0.095      8.399      0.000       0.611       0.982
date_tuple           -0.0015      0.000     -3.681      0.000      -0.002      -0.001
=====================================================================================
```

```
In [18]: X = df1[['tags','quality','colorful','face',"description_exist","date_tuple"]]
         Y = df1['views']
         poisson = sm.Poisson(Y, X).fit()
         print(poisson.summary())

Optimization terminated successfully.
         Current function value: 1515.536377
         Iterations 16
                          Poisson Regression Results
==============================================================================
Dep. Variable:                  views   No. Observations:                  265
Model:                        Poisson   Df Residuals:                      258
Method:                           MLE   Df Model:                            6
Date:                Mon, 14 Nov 2022   Pseudo R-squ.:                  0.2377
Time:                        11:06:24   Log-Likelihood:            -4.0162e+05
converged:                       True   LL-Null:                   -5.2686e+05
Covariance Type:            nonrobust   LLR p-value:                     0.000
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
tags                  0.0275   9.46e-05    290.912      0.000       0.027       0.028
NonCommercial         5.6588      0.008    721.775      0.000       5.643       5.674
quality            1.237e-05   1.01e-06     12.236      0.000    1.04e-05    1.44e-05
colorful              0.0126   7.54e-05    167.825      0.000       0.013       0.013
face                  0.3775      0.003    121.185      0.000       0.371       0.384
description_exist     1.0540      0.005    201.436      0.000       1.044       1.064
date_tuple           -0.0030   1.98e-05   -153.613      0.000      -0.003      -0.003
=====================================================================================
```

```
In [19]: X = df[['tags','quality','colorful','face',"description_exist","date_tuple"]]
         Y = df['comments']
         poisson = sm.Poisson(Y, X).fit()
         print(poisson.summary())

Optimization terminated successfully.
         Current function value: 2.124472
         Iterations 7
                          Poisson Regression Results
==============================================================================
Dep. Variable:               comments   No. Observations:                  449
Model:                        Poisson   Df Residuals:                      443
Method:                           MLE   Df Model:                            5
Date:                Mon, 14 Nov 2022   Pseudo R-squ.:                  0.1096
Time:                        11:06:37   Log-Likelihood:                -953.89
converged:                       True   LL-Null:                       -1071.3
Covariance Type:            nonrobust   LLR p-value:                 9.947e-49
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
tags                  0.0019      0.004      0.501      0.617      -0.006       0.009
quality              -0.0005   6.72e-05     -6.870      0.000      -0.001      -0.000
colorful              0.0126      0.002      7.559      0.000       0.009       0.016
face                  0.0863      0.097      0.887      0.375      -0.104       0.277
description_exist     0.6174      0.114      5.423      0.000       0.394       0.841
date_tuple           -0.0042      0.000    -10.683      0.000      -0.005      -0.003
=====================================================================================
```

Foundation

```
In [15]: X = df2[['tags','quality','colorful','face',"description_exist","date_tuple"]]
         Y = df2['favorites']
         poisson = sm.Poisson(Y, X).fit()
         print(poisson.summary())

Optimization terminated successfully.
         Current function value: 0.837019
         Iterations 9
                          Poisson Regression Results
==============================================================================
Dep. Variable:              favorites   No. Observations:                  106
Model:                        Poisson   Df Residuals:                      100
Method:                           MLE   Df Model:                            5
Date:                Sun, 13 Nov 2022   Pseudo R-squ.:                  0.6385
Time:                        23:54:13   Log-Likelihood:                -88.724
converged:                       True   LL-Null:                       -245.42
Covariance Type:            nonrobust   LLR p-value:                 1.315e-65
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
tags                  0.0590      0.007      8.297      0.000       0.045       0.073
quality               0.0005   8.25e-05      6.205      0.000       0.000       0.001
colorful             -0.0320      0.006     -5.469      0.000      -0.043      -0.021
face                 -0.4072      0.240     -1.694      0.090      -0.878       0.064
description_exist     2.1359      0.224      9.519      0.000       1.696       2.576
date_tuple           -0.0670      0.008     -8.888      0.000      -0.082      -0.052
=====================================================================================
```

```
In [21]: X = df2[['tags','quality','colorful','face',"description_exist","date_tuple"]]
         Y = df2['views']
         poisson = sm.Poisson(Y, X).fit()
         print(poisson.summary())

Optimization terminated successfully.
         Current function value: 449.364307
         Iterations 10
                          Poisson Regression Results
==============================================================================
Dep. Variable:                  views   No. Observations:                  106
Model:                        Poisson   Df Residuals:                       99
Method:                           MLE   Df Model:                            6
Date:                Mon, 14 Nov 2022   Pseudo R-squ.:                  0.6681
Time:                        11:07:32   Log-Likelihood:                 -47633.
converged:                       True   LL-Null:                   -1.4353e+05
Covariance Type:            nonrobust   LLR p-value:                     0.000
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
tags                  0.0521      0.000    243.201      0.000       0.052       0.053
NonCommercial         5.6396      0.012    454.613      0.000       5.615       5.664
quality               0.0002   2.24e-06     71.535      0.000       0.000       0.000
colorful             -0.0138      0.000    -91.932      0.000      -0.014      -0.013
face                 -0.1875      0.006    -33.095      0.000      -0.199      -0.176
description_exist     2.4923      0.009    286.492      0.000       2.475       2.509
date_tuple           -0.0222   9.88e-05   -224.345      0.000      -0.022      -0.022
=====================================================================================
```

```
In [22]: X = df2[['tags','quality','colorful','face',"description_exist","date_tuple"]]
         Y = df2['comments']
         poisson = sm.Poisson(Y, X).fit()
         print(poisson.summary())

Optimization terminated successfully.
         Current function value: 1.105859
         Iterations 9
                          Poisson Regression Results
==============================================================================
Dep. Variable:               comments   No. Observations:                  106
Model:                        Poisson   Df Residuals:                      100
Method:                           MLE   Df Model:                            5
Date:                Mon, 14 Nov 2022   Pseudo R-squ.:                  0.6378
Time:                        11:07:45   Log-Likelihood:                -117.22
converged:                       True   LL-Null:                       -323.60
Covariance Type:            nonrobust   LLR p-value:                 5.261e-87
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
tags                  0.0391      0.010      3.973      0.000       0.020       0.058
quality               0.0006      0.000      4.269      0.000       0.000       0.001
colorful             -0.0512      0.008     -6.754      0.000      -0.066      -0.036
face                 -1.4818      0.355     -4.174      0.000      -2.178      -0.786
description_exist     3.8128      0.218     17.477      0.000       3.385       4.240
date_tuple           -0.0809      0.011     -7.509      0.000      -0.102      -0.060
=====================================================================================
```

- **Testing**

Lipstick

In [19]:
```
X = df1[['tags','quality','colorful','face',"description_exist","date_tuple"]]
Y = (df1['favorites']-df1['favorites'].min())/(df1['favorites'].max()-df1['favorites'].min())
poisson_training_results = sm.GLM(Y, X, family=sm.families.Poisson()).fit()
print(poisson_training_results.summary())
```

In [20]:
```
X_test = df_test1[['tag','quality','colorful','face','Description','date_tuple']]
Y_test = (df_test1["like"]-df_test1['like'].min())/(df_test1['like'].max()-df_test1['like'].min())
poisson_predictions = poisson_training_results.get_prediction(X_test)
#summary_frame() returns a pandas DataFrame
predictions_summary_frame = poisson_predictions.summary_frame()
print(predictions_summary_frame)
```

In [21]:
```
predicted_counts = predictions_summary_frame['mean']
actual_counts = Y_test
```

In [17]:
```
predictions_summary_frame['rank'] = predictions_summary_frame["mean"].rank(axis=0,method='min',ascending=True)
df_test1["rank"] = df_test1["like"].rank(axis=0,method='min',ascending=True)
```

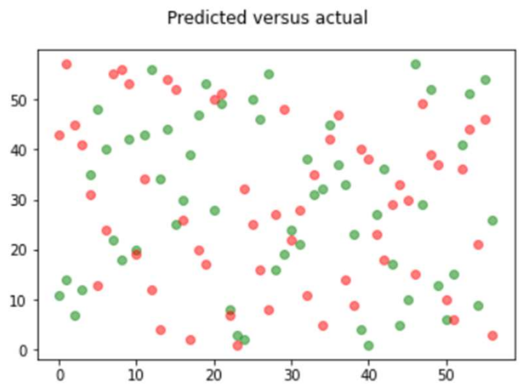
In [29]:
```
predictions_summary_frame['rank'] = predictions_summary_frame["mean"].rank(axis=0,method='min',ascending=True)
df_test1["rank"] = Y_test.rank(axis=0,method='min',ascending=True)
```

In [32]:
```
import scipy.stats as ss

data1 = Y_test.values.tolist()
data2 = predictions_summary_frame['mean'].values.tolist()

stat, p = ss.ranksums(data1, data2)
print(stat,p)
```

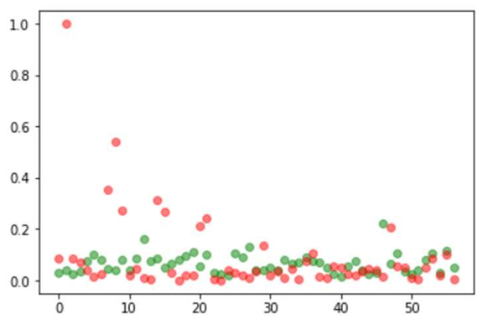
```
-2.553063651338985 0.010677998035961233
```

In [33]:
```
from matplotlib import pyplot as plt
```

In [34]:
```
fig = plt.figure()
fig.suptitle('Predicted versus actual')
predicted = plt.plot(df_test1.index, predictions_summary_frame["rank"], 'go', label='Predicted counts', alpha=0.5)
actual = plt.plot(df_test1.index, df_test1["rank"], 'ro', label='Actual counts', alpha=0.5)
plt.show()
```



```
In [36]: fig = plt.figure()
         fig.suptitle('Predicted versus actual')
         predicted = plt.plot(df_test1.index, predictions_summary_frame["mean"], 'go', label='Predicted counts', alpha=0.5)
         actual = plt.plot(df_test1.index, Y_test, 'ro', label='Actual counts', alpha=0.5)
         plt.show()
```



Foundation

```
In [37]: X = df2[['tags','quality','colorful','face',"description_exist","date_tuple"]]
         Y = (df2['favorites']-df2['favorites'].min())/(df2['favorites'].max()-df2['favorites'].min())
         poisson_training_results = sm.GLM(Y, X, family=sm.families.Poisson()).fit()
         print(poisson_training_results.summary())
```

...

```
In [38]: X_test = df_test2[['tag','quality','colorful','face','Description','date_tuple']]
         Y_test = (df_test2["like"]-df_test2['like'].min())/(df_test2['like'].max()-df_test2['like'].min())
         poisson_predictions = poisson_training_results.get_prediction(X_test)
         predictions_summary_frame = poisson_predictions.summary_frame()
         print(predictions_summary_frame)
```

...

```
In [39]: import scipy.stats as ss

         data1 = Y_test.values.tolist()
         data2 = predictions_summary_frame['mean'].values.tolist()

         stat, p = ss.ranksums(data1, data2)
         print(stat,p)

         2.9613040914970745 0.0030633931339710003
```

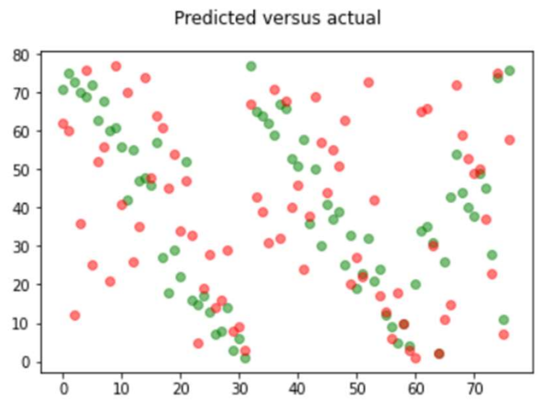
```
In [40]: predicted_counts = predictions_summary_frame['mean']
         actual_counts = Y_test
```

```
In [41]: predictions_summary_frame['rank'] = predictions_summary_frame["mean"].rank(axis=0,method='min',ascending=True)
         df_test2["rank"] = Y_test.rank(axis=0,method='min',ascending=True)
```

In [42]:
```
delta = predictions_summary_frame['rank'] - df_test2["rank"]
change = delta**2
print(change.sum())
```

30935.0

In [43]:
```
fig = plt.figure()
fig.suptitle('Predicted versus actual')
predicted = plt.plot(df_test2.index, predictions_summary_frame["rank"], 'go', label='Predicted counts', alpha=0.5)
actual = plt.plot(df_test2.index, df_test2["rank"], 'ro', label='Actual counts', alpha=0.5)
plt.show()
```



In [44]:
```
fig = plt.figure()
fig.suptitle('Predicted versus actual')
predicted = plt.plot(df_test2.index, predictions_summary_frame["mean"], 'go', label='Predicted counts', alpha=0.5)
actual = plt.plot(df_test2.index, Y_test, 'ro', label='Actual counts', alpha=0.5)
plt.show()
```