

## ▼ 1 Do hot naked selfies get the most likes on Instagram?

### 1.1 A case study

My favorite instagrammer once said something along the lines of "I have thousands of professional modeling photos, but all you guys want to see is mirror selfies of me in my underwear."

Is this true? Would the average viewer prefer a candid lingerie photo over a glitzy shot on set with hair, makeup, wardrobe, etc? I decided to find out.

#### 1.1.1 Imports

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import urllib.request
from InstagramAPI import InstagramAPI
```

#### ▼ 1.1.2 Instagram API

Shout out to LevPasha on GitHub for writing these APIs. He did all the hard work.

```
In [2]: api = InstagramAPI('username', 'password')
```

Login to my account.

```
In [3]: api.login()

Request return 405 error!
{'message': '', 'status': 'fail'}
Login success!
```

```
Out[3]: True
```

### ▼ 1.1.3 Instagrammer Selection

Obviously to do this in real depth would require pulling data from at least hundreds of users. That way you can control for differences between a given user's following and with more data points the trends found tend to be more extrapolatable. For example, different cultures have different beauty standards. People will follow different versions of what they consider attractive on Instagram and this will ripple over into which photos get the most likes and comments. Thus only looking at a single user won't necessarily provide insight into greater trends. However, I don't have that kind of free time.

My crush du jour is [Teddy Quinlivan \(https://www.instagram.com/teddy\\_quinlivan/\)](https://www.instagram.com/teddy_quinlivan/), so we'll be using her account as the data set. She has over 1000 photos ranging from walking the Louis Vuitton runway to pulling lewks at China Chalet to random art finds. There's a lot of hot content on her account, but there's also a plethora of content that might disinterest the typical male [@emrata \(https://www.instagram.com/emrata/\)](https://www.instagram.com/emrata/) follower. Hopefully this will give us enough variation to see if the hot selfies really do get all the likes.

Using the API we'll search for her account by handle and find her account number.

```
In [4]: api.searchUsername('teddy_quinlivan')
api.LastJson
```

```
Out[4]: {'user': {'pk': 1904001441,
  'username': 'teddy_quinlivan',
  'full_name': 'Theodora Quinlivan',
  'is_private': False,
  'profile_pic_url': 'https://scontent-lga3-1.cdninstagram.com/v/t51.2885-19/s150x150/47695016_2286338751596772_7740193201025187840_n.jpg?_nc_ht=scontent-lga3-1.cdninstagram.com&_nc_ohc=MTnYvmAYn6AAX8yN19a&oh=bb2a4a7887109b66f738e0a5545acc98&oe=5E928187',
  'profile_pic_id': '1953616360206528395_1904001441',
  'is_verified': True,
  'has_anonymous_profile_picture': False,
  'media_count': 1178,
  'geo_media_count': 0,
  'follower_count': 122596,
  'following_count': 769,
  'following_tag_count': 0,
  'biography': 'Live Fast, Die Last \n👠🎧🔪🌹👑\nVision - Los Angeles\nSelect - London\nPremium - Paris\nMonster - Milan\nSupreme - NYC',
  'biography_with_entities': {'raw_text': 'Live Fast, Die Last \n👠🎧🔪🌹👑\nVision - Los Angeles\nSelect - London\nPremium - Paris\nMonster - Milan\nSupreme - NYC'}}
```

### ▼ 1.1.4 Scraping the data

Now that we've found her account number we can get her entire feed and all meta data with a single command.

```
In [5]: user_data = api.getTotalUserFeed('1904001441')
```

From the raw data I'd like to collect 5 features: the timestamp, number of likes, number of comments, the caption, and location.

The API grabs the data with the most recent post as 0, so we'll reverse iterate through it so we can save the photos numbered from 0 being the first photo she ever posted to n being the most recent. I've placed all the feature grabbing lines in try/excepts because I found some photos had comments disabled or no location data as the feature was not part of the original app. For posts without this data 0/nulls were inserted into the data set.

I made a TQ\_Data folder on my desktop which will hold all of her photos numbered by post and a csv containing the metadata. The index of the csv will match the photo number.

```
In [6]: columns = ['timestamp', 'likes', 'comments', 'caption', 'location']
ar = np.zeros((len(user_data), len(columns)))
df = pd.DataFrame(ar, columns=columns)
print(len(user_data))
for i in range(len(user_data) - 1, -1, -1): # loop backwards so first photo is 0 and
    try:
        time = user_data[i]['taken_at']
    except:
        time = ''
    try:
        likes = user_data[i]['like_count']
    except:
        likes = 0
    try:
        comments = user_data[i]['comment_count'] # comments may be disabled
    except:
        comments = 0
    try:
        caption = user_data[i]['caption']['text']
    except:
        caption = ''
    try:
        location = user_data[i]['location']['name'] # location wasn't always a feature
    except:
        location = ''

    df.loc[len(user_data) - i - 1, :] = [time, likes, comments, caption, location]

    try:
        link = TQ_data[i]['image_versions2']['candidates'][0]['url']
        urllib.request.urlretrieve(link, f'~/Desktop/TQ_Data/TQ_{len(TQ_data) - i - 1}')
    except:
        try:
            link = TQ_data[i]['carousel_media'][1]['image_versions2']['candidates'][0]
            urllib.request.urlretrieve(link, f'~/Desktop/TQ_Data/TQ_{len(TQ_data) - i - 1}')
        except:
            pass

df.to_csv('~/Desktop/TQ_Data/TQ_Data.csv')
```

1177

## ▼ 1.2 Data Labeling

The funny thing about machine learning is that the initial steps are pretty human intensive. Before computers and data scientists can do anything with image data the data has to be labeled. Humans know cats vs dogs by sight, but a computer has to relate it to the 3 letter word c-a-t.

That means I went through over 1000 photo manually labeling features: rating, showing, taken\_by.

Let's break this down, because this is the most controversial/important part. Classifying images is subjective, so I made the decision that there would be new features for each images defined as such:

**Rating:** a measure of attractiveness

- Casual: a photo of the user in a normal day-to-day context
- Glamour Shot: a photo that evidences prior setup (eg. a fashion show or something clearly not candid)
- Hot: a photo of the user clothed where it seems a more than usual effort was made to look attractive
- Not Self: a photo of anything other than the user (may include other people, objects, drawings, etc)
- Nude: a photo where the user is naked
- Scantily Clad: a photo where the user is wearing highly revealing clothing

**Showing:** a measure how much of the user's body is shown

- Body: a photo containing either the entirety or majority of the user's body
- Bust: a photo of the user from the bust up
- Head: a photo of the user's head
- null: photos not of the user are marked null

**Taken By:** tagging the photographer

- Amateur: a photo taken by any non-professional
- Professional: a photo taken by what appears to be a career photographer
- Self: a photo taken by the user - a selfie in the modern parlance
- null: photos not of the user are marked null

### 1.2.1 Subjectivity

The choosing of these features and each one of their facets is a highly subjective matter. One could argue that just by the items chosen I have already skewed that data and introduced a bias. Probably. I acknowledge there are better ways to do this, for example conducting preliminary studies to see how other people categorize the images or having multiple people label the photos to control for personal preferences.

I'm doing this for fun, so that would have been a lot more than I bargained for. It just means that the results in this study are dependant on the labels used and the account chosen. Whether the conclusions in here map over to the rest of Instagram... who knows. Maybe another day I or another may take this up again and conduct it with more scientific rigor.

### 1.2.2 Examples

Before going on, it will be helpful to see an example of each of the categories defined above. First let's make a function to easily plot images by their number.

```
In [116]: def plot_img(num):
            plt.figure(figsize=(5, 5))
            plt.axis('off')
            plt.imshow(mping.imread(f'/Users/XXI/Desktop/TQ_Data/TQ_{num}.jpg'));
```

### ▼ 1.2.3 casual, head, self

This casual selfie with a friend only includes her head.

```
In [117]: plot_img(477)
```



### ▼ 1.2.4 glamour shot, body, professional

The fashion brand name across the photo makes it obvious that this photo has been staged and was shot by a professional. We can see Teddy's entire body here.

```
In [118]: plot_img(893)
```



### ▼ 1.2.5 glamour shot, body, amateur

Here Teddy is clearly wearing a couture ensemble and her whole body is visible. However, this photo is not on a set and likely shot by a friend.

```
In [119]: plot_img(410)
```



### ▼ 1.2.6 hot, bust, amateur

Between the makeup, jewelry, and spaghetti strap top I think it's safe to say she didn't roll out of bed like this. The image is framed from slightly below the shoulders and up and most likely not taken by a professional.

```
In [120]: plot_img(1079)
```



### ▼ 1.2.7 scantily clad, body, amateur

In this photo there is no guesswork. Just read the caption:

"Ok so last night I was invited to this **messy lingerie themed** Halloween party, and of course I'm so down because I'm basically a hooker and an attention whore so this party was right up my ally. I didn't end up having any fun cuz they make you sign an NDA at the door and it was a pussy fest (there were only girls there). Not only that, but I'm so used to being the only tall hot slut at parties, and this particular party was filled with so many hot girls who had way better bodies than me, so I didn't even stand out or feel special, I was just 1 in 1000 sexy hot slutty girls. I tried to take this picture in the bathroom to discreetly capture my glam and I would like to **thank the real MVP @giselleadorabrune (<https://www.instagram.com/giselleadorabrune/>) for Kneeling in a pile of toilette paper to take this picture\*** of me. Halloween parties are a overrated scam, next year I'm going as a ghost."

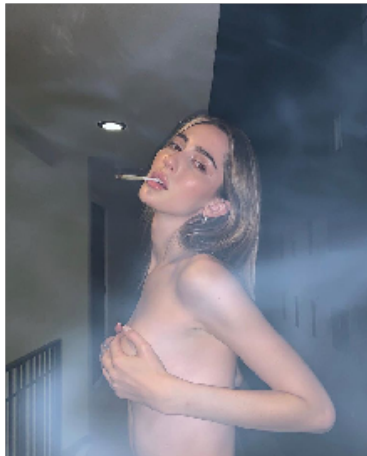
```
In [121]: plot_img(1125)
```



### ▼ 1.2.8 nude, bust, amateur

This is about as naked as you can get on Instagram.

```
In [122]: plot_img(1137)
```



▼ 1.2.9 not self, null, null

For this post Teddy used a meme for the image.

```
In [123]: plot_img(1151)
```



```
In [99]: data = pd.read_csv('/Users/XXI/Desktop/TQ_Data/TQ_Data.csv', header=0, low_memory=False)
display(data.head())
```



	photo_num	timestamp	likes	comments	caption	location	rating	showing	taken_by
0	1176	1583440848	7980	79	Ph: @attilakiss.photo	Los Angeles, California	not self	NaN	NaN
1	1175	1583342712	7631	105	🦀🦀🦀	Pirate's Cove Beach	glamour shot	body	amateur
2	1174	1583181388	2462	28	A whole ass mood	Los Angeles, California	not self	NaN	NaN
3	1173	1583005859	9884	182	🦀🦀🦀	Los Angeles, California	glamour shot	body	amateur
4	1172	1582110737	1120	14	Terminator 2: Judgement Day (1991)	Times Square, New York City	not self	NaN	NaN

```
In [21]: data.fillna('', inplace=True)
print(data['rating'].unique())
print(data['showing'].unique())
print(data['taken_by'].unique())

['not self' 'glamour shot' 'hot' 'nude' 'casual' 'scantily clad' '0']
['' 'body' 'bust' 'head']
['' 'amateur' 'self' 'professional']
```



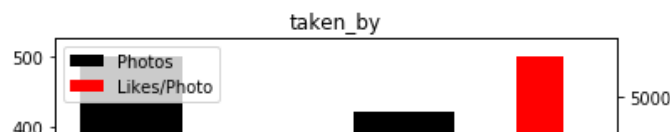
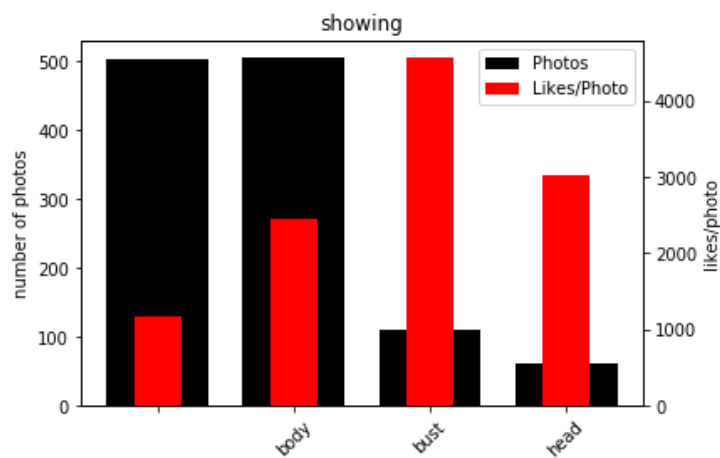
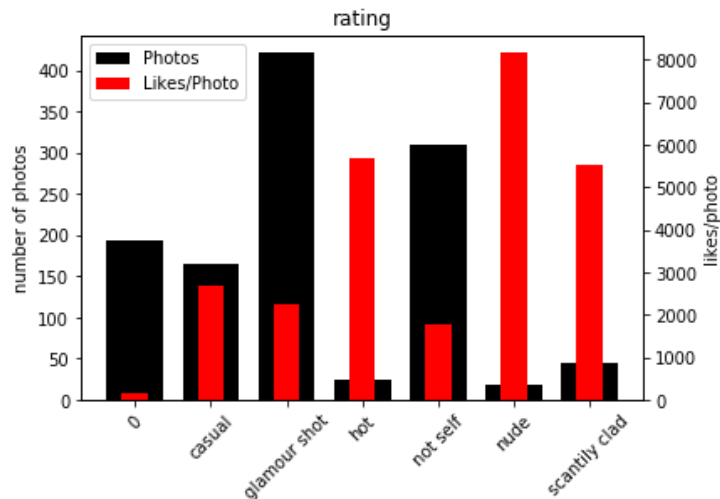
```
In [92]: data['datetime'] = pd.to_datetime(data['timestamp'], unit='s')
data['date'] = data['datetime'].dt.date
data['day'] = data['datetime'].dt.dayofweek
data['hour'] = data['datetime'].dt.strftime('%H').astype(int)
display(data.head())
```

	photo_num	timestamp	likes	comments	caption	location	rating	showing	taken_by	datetime
0	1176	1583440848	7980	79	@atilkiss.photo	Ph: Los Angeles, California	not self	NaN	NaN	2020-03-05 20:40:48
1	1175	1583342712	7631	105		Pirate's Cove Beach	glamour shot	body	amateur	2020-03-04 17:25:12
2	1174	1583181388	2462	28	A whole ass mood	Los Angeles, California	not self	NaN	NaN	2020-03-02 20:36:28
3	1173	1583005859	9884	182		Los Angeles, California	glamour shot	body	amateur	2020-02-29 19:50:59
4	1172	1582110737	1120	14	Terminator 2: Judgement Day (1991)	Times Square, New York City	not self	NaN	NaN	2020-02-19 11:12:17

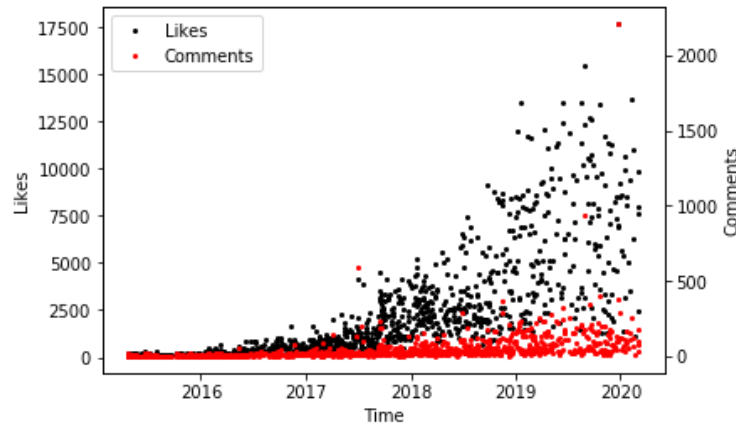
```
In [86]: features = ['rating', 'showing', 'taken_by']
def bar_category(category):
    ratings_x = np.sort(data[category].unique())
    ratings_y = []
    ratings_likes = []
    for i in ratings_x:
        ratings_y.append(len(data.loc[data[category] == i]))
        ratings_likes.append(sum(data.loc[data[category] == i]['likes'])/ratings_y[-1])

    fig, ax = plt.subplots()
    plt1 = ax.bar(ratings_x, ratings_y, width=0.75, color='k')
    ax0 = ax.twinx()
    plt2 = ax0.bar(ratings_x, ratings_likes, width=0.35, color='r')
    ax.set_xticklabels(ratings_x, rotation=45)
    ax.set_ylabel('number of photos')
    ax0.set_ylabel('likes/photo')
    plts = [plt1, plt2]
    labels = ['Photos', 'Likes/Photo']
    ax.set_title(category)
    ax0.legend(plts, labels);

for i in features:
    bar_category(i)
```

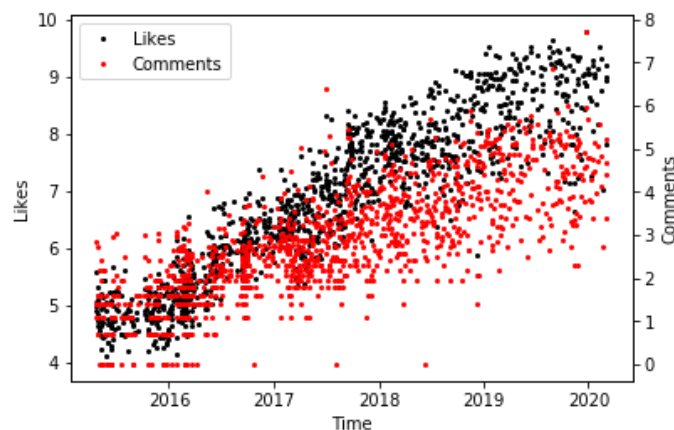


```
In [286]: fig, ax = plt.subplots()
plt1 = ax.plot_date(data['date'], data['likes'], color='k', markersize=2)
ax0 = ax.twinx()
plt2 = ax0.plot_date(data['date'], data['comments'], color='r', markersize=2)
ax.set_xlabel('Time')
ax.set_ylabel('Likes')
ax0.set_ylabel('Comments')
plt1 + plt2
labels = ['Likes', 'Comments']
ax.legend(plt1, labels);
```



```
In [294]: data0 = data.copy()
data0['likes'] = np.log(data0['likes'] + 0.1)
data0['comments'] = np.log(data0['comments'] + 1)
```

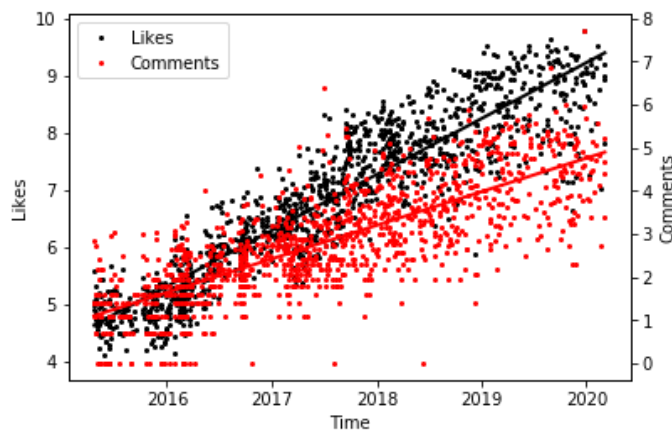
```
In [295]: fig, ax = plt.subplots()
plt1 = ax.plot_date(data0['date'], data0['likes'], color='k', markersize=2)
ax0 = ax.twinx()
plt2 = ax0.plot_date(data0['date'], data0['comments'], color='r', markersize=2)
ax.set_xlabel('Time')
ax.set_ylabel('Likes')
ax0.set_ylabel('Comments')
plt1 + plt2
labels = ['Likes', 'Comments']
ax.legend(plt1, labels);
```



```
In [296]: from scipy.stats import linregress
linreg_likes = linregress(data0['timestamp'], data0['likes'])
linreg_comments = linregress(data0['timestamp'], data0['comments'])
print(linreg_likes)
print(linreg_comments)
```

```
LinregressResult(slope=3.098174857200752e-08, intercept=-39.658303444663126, rvalue=0.9159308198647161, pvalue=0.0, stderr=3.9603415024524195e-10)
LinregressResult(slope=2.4673588036132233e-08, intercept=-34.17753450641675, rvalue=0.776072145481548, pvalue=1.672274837537843e-237, stderr=5.849188815758522e-10)
```

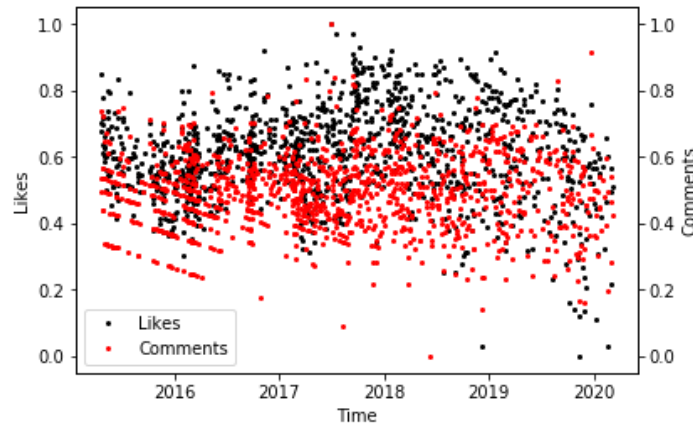
```
In [297]: fig, ax = plt.subplots()
plt1 = ax.plot_date(data0['date'], data0['likes'], color='k', markersize=2)
ax.plot_date(data0['date'], data0['timestamp']*linreg_likes[0] + linreg_likes[1], color='k', markersize=2)
ax0 = ax.twinx()
plt2 = ax0.plot_date(data0['date'], data0['comments'], color='r', markersize=2)
ax0.plot_date(data0['date'], data0['timestamp']*linreg_comments[0] + linreg_comments[1], color='r', markersize=2)
ax.set_xlabel('Time')
ax.set_ylabel('Likes')
ax0.set_ylabel('Comments')
plt1s = plt1 + plt2
labels = ['Likes', 'Comments']
ax.legend(plt1s, labels);
```



```
In [298]: data0['likes'] -= data0['timestamp']*linreg_likes[0] + linreg_likes[1]
data0['comments'] -= data0['timestamp']*linreg_comments[0] + linreg_comments[1]

data0['likes'] = (data0['likes'] - min(data0['likes']))/(max(data0['likes']) - min(data0['likes']))
data0['comments'] = (data0['comments'] - min(data0['comments']))/(max(data0['comments']) - min(data0['comments']))
```

```
In [299]: fig, ax = plt.subplots()
plt1 = ax.plot_date(data0['date'], data0['likes'], color='k', markersize=2)
ax0 = ax.twinx()
plt2 = ax0.plot_date(data0['date'], data0['comments'], color='r', markersize=2)
ax.set_xlabel('Time')
ax.set_ylabel('Likes')
ax0.set_ylabel('Comments')
plt1 = plt1 + plt2
labels = ['Likes', 'Comments']
ax.legend(plt1, labels);
```

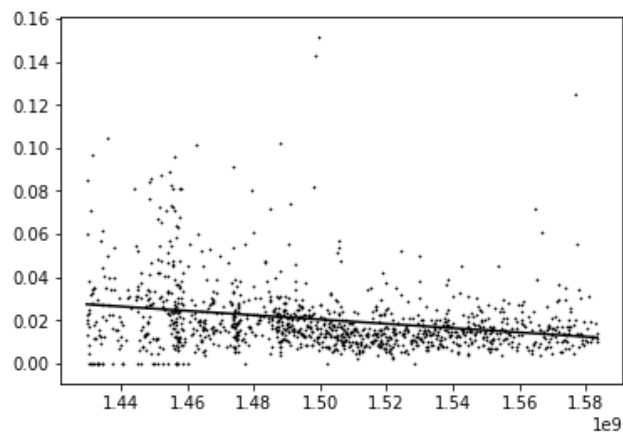


```
In [300]: display(data0.loc[data0['likes'] >= 0.7].head()) # [['rating', 'showing', 'taken_by
```

	photo_num	timestamp	likes	comments	caption	location	rating	showing	taken_by	date
24	1152	1577042633	0.756491	0.913261	I've been called a slut and a whore more times...	NO FUCKS GIVEN	nude	bust	amateur	2019-19:
54	1122	1571590005	0.729885	0.692534	What's Crackin?	Beverly Hills, California	casual	bust	amateur	2019-16:
70	1106	1569004988	0.735018	0.611344	♥ SAVAGE X FENTY ♥ I told myself I wouldn't wa...	Barclay Center, Brooklyn New York	glamour shot	body	professional	2019-18:
74	1102	1568395575	0.741527	0.680845	👑 PAPER MAGAZINE 👑 How fucking fabulous to get ...	Buckingham Palace	glamour shot	bust	professional	2019-17:
81	1095	1567095775	0.744366	0.593289	I moved to Paris 1 year ago because I had a qu...	Paris, France	glamour shot	body	amateur	2019-16:

```
In [301]: linreg_engagement = linregress(data['timestamp'], data['comments']/(data['likes'] +
print(linreg_engagement)
plt.scatter(data['timestamp'], data['comments']/data['likes'], c='k', s=0.5)
plt.plot(data['timestamp'], data['timestamp']*linreg_engagement[0] + linreg_engagem
```

LinregressResult(slope=-1.0024073427571824e-10, intercept=0.17068688613319288, rvalue=-0.2515115024966806, pvalue=1.939566221117996e-18, stderr=1.1253236401958903e-11)



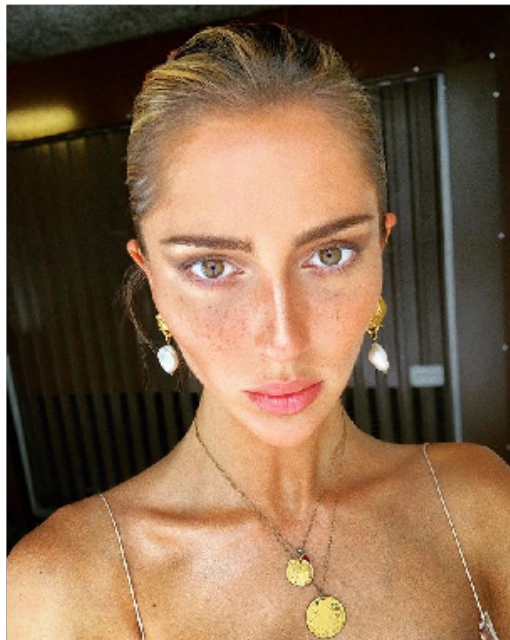
```
In [276]: print(max(data['likes']))
print(data0['likes'].mean())
n = 1089
display(data.loc[data['photo_num'] == n])
display(data0.loc[data0['photo_num'] == n])
# print(data0.loc[data0.loc[data0['photo_num'] == n].index[0], 'caption'])
plot_img(n)
```

17682

0.6118157385168325

	photo_num	timestamp	likes	comments	caption	location	rating	showing	taken_by	datetime	
87	1089	1565885669	11357	201	No lie, One time a guy offered to pay me \$500 ...	Area 51	casual	head	self	2019-08-15 16:14:29	2019-0

	photo_num	timestamp	likes	comments	caption	location	rating	showing	taken_by	datetime	
87	1089	1565885669	0.732245	0.616029	No lie, One time a guy offered to pay me \$500 ...	Area 51	casual	head	self	2019-08-15 16:14:29	201



```
In [ ]: ##### print(data['comments'].idxmax())
data.loc[data['comments'].idxmin(), :]
# # print(len(data))
# print(min(data['likes']))
# TeddyQ correlation
```

```
In [131]: plot_img(852)
```



```
In [117]: data.loc[data['likes'] < 500].head()
```

Out[117]:

	photo_num	timestamp	likes	comments	caption	location	rating	showing	taken_by
208	968	1544459378	353	3	REDKEN ❤️ \n#colorgelslacquers #redken @redken		not self		
452	724	1512403703	460	4	Music: Julian Winding - Disconnected	New York, New York	0		
540	636	1504047935	495	9	Drew Barrymore - Scream 1996	New York, New York	0		
545	631	1503358512	477	6	Ghost in the Shell - 1995	New York, New York	0		
555	621	1502471017	423	6	ああ助かった、金曜日だ	New York, New York	0		



In [122]:

```
plot_img(1152)
```



In [108]:

```
plot_img(1144)
```



In [113]: `plot_img(37)`



In [ ]: