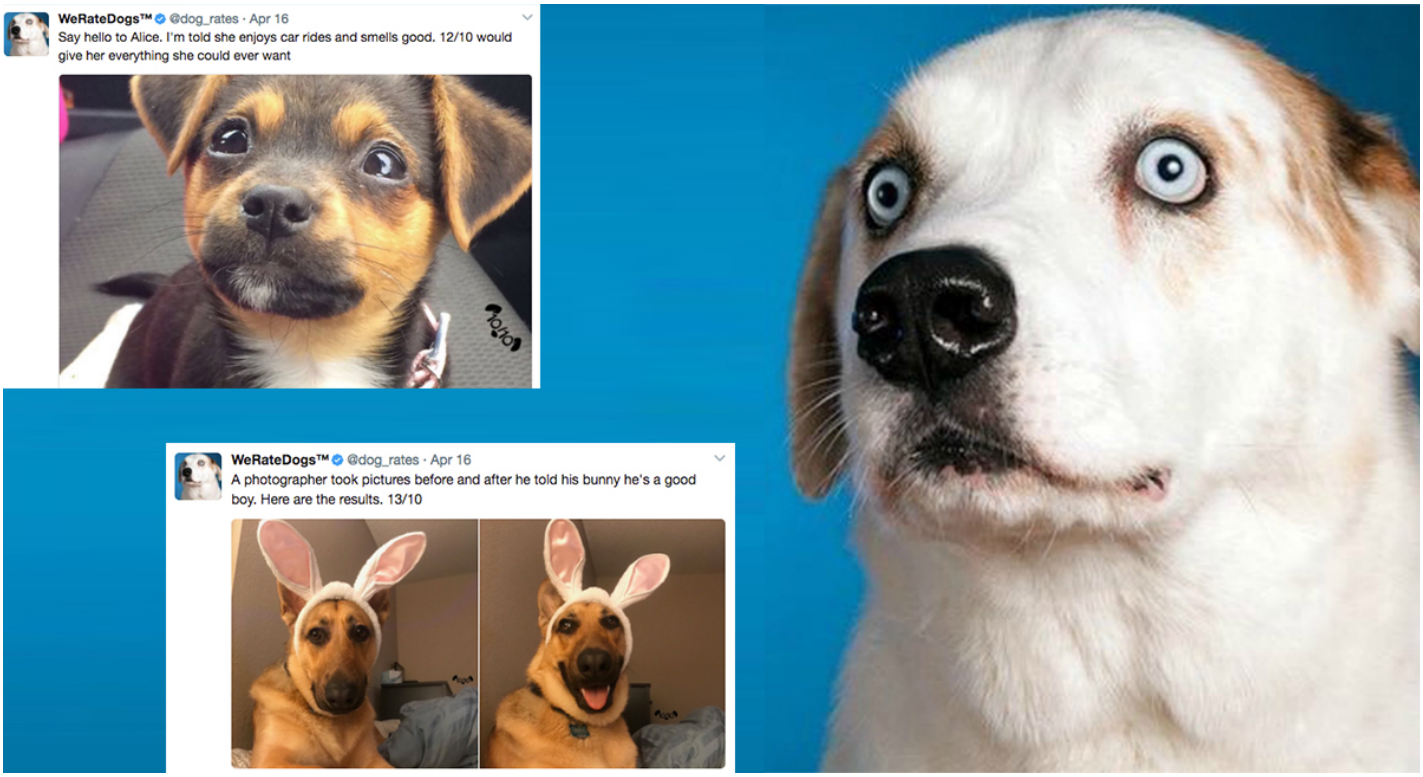
本数据集收集的是一个推特博主叫WeRateDog的每条推特。（超级有意思的博主）这个博主就是对狗狗打分 传送门->https://twitter.com/dog_rates



** 我在做的事情就是 **

- 1.数据进行收集
- 2.评估数据质量问题和清洁度（结构）问题，
- 3.清洗数据集 ☺

收集

```
# 导入需要的库
import numpy as np
from aip import AipImageClassify
from bs4 import BeautifulSoup
import requests
import re
import pandas as pd
```

提示：你需要收集的文件有：

1. 收集手头文件 `twitter_archive_enhanced.csv`，其中包含了一些主要的推特信息，是本次清洗的主要数据，其中的评分、地位和名字等数据是从 `text` 原文中提取的，但是提取的并不好，评分并不都是正确的，狗的名字和地位也有不正确的 。**如果你想用评分、地位和名字进行分析和可视化，需要评估和清洗这些列。完成这些列的评估和清洗，你可以学到更加实用的技能。**
2. 编程下载收集互联网文件：`image-predictions.tsv`，其中包含了推特图像预测信息，根据推特中的图片预测出狗狗种类；
3. 查询 API 收集额外推特信息 `tweet_json.txt`，如果你无法访问 Twitter 的话，可以直接读取项目可供下载的 `tweet_json.txt` 文件，从中提取所需数据。至少需要提取转发数（`retweet_count`）和喜欢数（`favorite_count`）这两列，**如果你的分析中不需要用到其他列，则不需要收集其他列。**如果提取了其他列只用于清洗，那么这样的清洗没有意义。

```
# 收集文件 1 保存为 twitter_archive_enhanced
# 读取数据
twitter_archive_enhanced = pd.read_csv('twitter-archive-enhanced.csv')

# 显示前两行
twitter_archive_enhanced.head(2)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text	retweeted_status_id	retweeted_status_user_id	retweeted

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text	retweeted_status_id	retweeted_status_user_id	retweeted
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56 +0000	<a href="http://twitter.com/download/iphone" r...	This is Phineas. He's a mystical boy. Only eve...	NaN	NaN	NaN
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	<a href="http://twitter.com/download/iphone" r...	This is Tilly. She's just checking pup on you....	NaN	NaN	NaN

```
# 收集文件 2 保存为 tweet_json
# 显示数据前2行
tweet_json = pd.read_json("tweet_json.json", lines = True)

# 显示前两行
tweet_json.head(2)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	contributors	coordinates	created_at	display_text_range	entities	extended_entities	favorite_count	favorited	full_text	geo	...	possibly_sensitive_appealable	quoted_status	quo
0	NaN	NaN	2017-08-01 16:23:56	[0, 85]	{'hashtags': [], 'symbols': [], 'user_mentions...	{'media': [{'id': 892420639486877696, 'id_str'...	39492	False	This is Phineas. He's a mystical boy. Only eve...	NaN	...	0.0	NaN	NaN
1	NaN	NaN	2017-08-01 00:17:27	[0, 138]	{'hashtags': [], 'symbols': [], 'user_mentions...	{'media': [{'id': 892177413194625024, 'id_str'...	33786	False	This is Tilly. She's just checking pup on you....	NaN	...	0.0	NaN	NaN

2 rows × 31 columns

```
# 收集文件 3 保存为 image_predictions
r = requests.get("https://raw.githubusercontent.com/udacity/new-dand-advanced-china/master/%E6%95%B0%E6%8D%AE%E6%B8%85%E6%B4%97/WeRateDogs%E9%A1%B9%E7%9B%AE/image-predictions.tsv")

# # 新建空的文件image-predictions_byPythonDownload.tsv
fileobj = open("image-predictions_byPythonDownload.tsv", 'wb')

# # 将数据写入fileobj中
fileobj.write(r.content)
fileobj.close()

image_predictions = pd.read_csv("image-predictions_byPythonDownload.tsv", sep = '\t')
image_predictions.head(5)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	tweet_id	jpg_url	img_num	p1	p1_conf	p1_dog	p2	p2_conf	p2_dog	p3	p3_c
0	666020888022790149	https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg	1	Welsh_springer_spaniel	0.465074	True	collie	0.156665	True	Shetland_sheepdog	0.061
1	666029285002620928	https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg	1	redbone	0.506826	True	miniature_pinscher	0.074192	True	Rhodesian_ridgeback	0.072
2	666033412701032449	https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg	1	German_shepherd	0.596461	True	malinois	0.138584	True	bloodhound	0.116
3	666044226329800704	https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg	1	Rhodesian_ridgeback	0.408143	True	redbone	0.360687	True	miniature_pinscher	0.222
4	666049248165822465	https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg	1	miniature_pinscher	0.560311	True	Rottweiler	0.243682	True	Doberman	0.154

```
# 合并
```

评估

目测评估

```
# 目测评估三个数据集
```

twitter_archive_enhanced数据中有大量的空值
twitter_archive_enhanced数据中有些列是不需要用的比如 in_reply_to_status_id

tweet_json数据中包含了大量的空值

image_predictions数据中对狗狗品种认定，但是给出了三种结果，取其中最高的即可。

编程评估

```
# 使用 pandas 的各种方法评估三个数据集，比如 info value_counts 等

# 你需要添加更多的 code cell 和 markdown cell 来完成所有编程评估
```

```
twitter_archive_enhanced.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
tweet_id                      2356 non-null int64
in_reply_to_status_id         78 non-null float64
in_reply_to_user_id           78 non-null float64
timestamp                     2356 non-null object
source                        2356 non-null object
text                          2356 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2297 non-null object
rating_numerator              2356 non-null int64
rating_denominator            2356 non-null int64
name                          2356 non-null object
doggo                         2356 non-null object
floofer                       2356 non-null object
pupper                        2356 non-null object
puppo                         2356 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

```
image_predictions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
tweet_id    2075 non-null int64
jpg_url     2075 non-null object
img_num     2075 non-null int64
p1          2075 non-null object
p1_conf     2075 non-null float64
p1_dog      2075 non-null bool
p2          2075 non-null object
p2_conf     2075 non-null float64
p2_dog      2075 non-null bool
p3          2075 non-null object
p3_conf     2075 non-null float64
p3_dog      2075 non-null bool
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 152.1+ KB
```

```
tweet_json.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2352 entries, 0 to 2351
Data columns (total 31 columns):
contributors                    0 non-null float64
coordinates                     0 non-null float64
created_at                      2352 non-null datetime64[ns]
display_text_range              2352 non-null object
entities                        2352 non-null object
extended_entities               2073 non-null object
favorite_count                  2352 non-null int64
favorited                       2352 non-null bool
full_text                       2352 non-null object
geo                             0 non-null float64
id                              2352 non-null int64
id_str                          2352 non-null int64
in_reply_to_screen_name         78 non-null object
in_reply_to_status_id           78 non-null float64
```

```
in_reply_to_status_id_str        78 non-null float64
in_reply_to_user_id              78 non-null float64
in_reply_to_user_id_str          78 non-null float64
is_quote_status                  2352 non-null bool
lang                             2352 non-null object
place                            1 non-null object
possibly_sensitive               2211 non-null float64
possibly_sensitive_appealable    2211 non-null float64
quoted_status                    28 non-null object
quoted_status_id                 29 non-null float64
quoted_status_id_str             29 non-null float64
retweet_count                    2352 non-null int64
retweeted                        2352 non-null bool
retweeted_status                 177 non-null object
source                           2352 non-null object
truncated                        2352 non-null bool
user                             2352 non-null object
dtypes: bool(4), datetime64[ns](1), float64(11), int64(4), object(11)
memory usage: 505.4+ KB
```

提示：

- 完成目测评估和编程评估之后，总结列出你发现的三个数据集中的所有问题；
- 每个问题都要有对应的一句话或几句话描述；
- 最终至少要包含 8 个质量问题和 2 个整洁度问题。

质量

`twitter_archive_enhanced` 表格

- in_reply_to_user_id 因为是回复id需要删除
- in_reply_to_status_id 因为是回复id需要删除
- 如果该条推特是回复别人的，需要删除这一条
- source 列应当只包含iphone web内容
- timestamp 列的时间数据不是datatime类型
- name列数据异常
- doggo floofer pupper puppo 列数据缺失
- rating_denominator列中有异常值（等于0）
- favorite_count 列中有几个NaN
- retweet_count 列中有几个NaN
- favorite_count和retweet_count列应当是int64类型

`tweet_json` 表格

- contributors等列空值

`image_predictions` 表格

整洁度

- doggo floofer pupper puppo 可以合并成一个列
- retweeted_status_id和retweeted_status_user_id保留一个

清理

提示：

- 清理数据集之前需要先备份数据集；
- 按照下面示例的结构：**定义-代码-测试**，对提出的每个问题进行清洗。

```
# 备份三个数据集
twitter_archive_enhanced.to_csv("twitter_archive_enhanced.csv")
tweet_json.to_csv("tweet_json.csv")
image_predictions.to_csv("image_predictions.csv")
```

问题描述一

定义

in_reply_to_user_id 有78条，这些数据是回复。将其删除。

代码

```
# 解决问题一的代码
df_temp = twitter_archive_enhanced[twitter_archive_enhanced["in_reply_to_user_id"].isnull()==False]
twitter_archive_enhanced = twitter_archive_enhanced.drop(index=df_temp.index)
```

测试

```
# 测试问题一是否正确清理完成
twitter_archive_enhanced.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2278 entries, 0 to 2355
Data columns (total 17 columns):
```

```
tweet_id                      2278 non-null int64
in_reply_to_status_id         0 non-null float64
in_reply_to_user_id           0 non-null float64
timestamp                     2278 non-null object
source                        2278 non-null object
text                          2278 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2274 non-null object
rating_numerator              2278 non-null int64
rating_denominator            2278 non-null int64
name                          2278 non-null object
doggo                         2278 non-null object
floofer                       2278 non-null object
pupper                        2278 non-null object
puppo                         2278 non-null object
dtypes: float64(4), int64(3), object(10)
memory usage: 320.3+ KB
```

问题描述二

定义

source列应当只包含iphone web等内容

代码

```
# 解决问题二的代码
# 重置index列
twitter_archive_enhanced.reset_index(drop=True, inplace=True)
# twitter_archive_enhanced
```

```
'''
    解析出每一个html标记语言中的内容
    <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>
    提取出Twitter for iPhone
'''
# html = twitter_archive_enhanced['source'][1]
# t = BeautifulSoup(html, 'lxml')
# t.a.contents

for i in range(len(twitter_archive_enhanced)):
    html = twitter_archive_enhanced.loc[i, 'source']
    try:
        t = BeautifulSoup(html, 'lxml') # html转为BeautifulSoup
        twitter_archive_enhanced.loc[i, 'source'] = t.a.contents[0]
    except:
        twitter_archive_enhanced.loc[i, 'source'] = html
# twitter_archive_enhanced['source']
```

测试

```
# 测试问题二是否正确清理完成
twitter_archive_enhanced['source'].unique()
```

```
array(['Twitter for iPhone', 'Twitter Web Client', 'Vine - Make a Scene',
       'TweetDeck'], dtype=object)
```

问题描述三

定义

timestamp 列的时间数据不是datatime类型

代码

```
twitter_archive_enhanced['timestamp'] = pd.to_datetime(twitter_archive_enhanced['timestamp'])
```

检测

```
twitter_archive_enhanced.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2278 entries, 0 to 2277
Data columns (total 17 columns):
tweet_id                      2278 non-null int64
in_reply_to_status_id         0 non-null float64
in_reply_to_user_id           0 non-null float64
timestamp                     2278 non-null datetime64[ns]
source                        2278 non-null object
text                          2278 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2274 non-null object
```

```
rating_numerator              2278 non-null int64
rating_denominator            2278 non-null int64
name                          2278 non-null object
doggo                         2278 non-null object
floofer                       2278 non-null object
pupper                        2278 non-null object
puppo                         2278 non-null object
dtypes: datetime64[ns](1), float64(4), int64(3), object(9)
memory usage: 302.6+ KB
```

问题描述四

定义

name列数据异常。根据text列修补

- 使用正则表达式匹配名字，并且名字第一个字应该是大写.
- 如果text中没有写出名字那么用nan代替

代码

```
# 引用 https://blog.csdn.net/u010606346/article/details/84778363
twitter_archive_enhanced['name'] = twitter_archive_enhanced['text'].str.extract(r'(?:This is|named|Meet|Say hello to|name is|Here we have|Here is)\s([A-Z][^\s.,]*)')
```

测试

```
twitter_archive_enhanced['name'].unique()
```

```
array(['Phineas', 'Tilly', 'Archie', 'Darla', 'Franklin', nan, 'Jax',
       'Zoey', 'Cassie', 'Koda', 'Bruno', 'Ted', 'Stuart', 'Oliver',
       'Jim', 'Zeke', 'Ralphus', 'Canela', 'Gerald', 'Jeffrey', 'Maya',
       'Mingus', 'Derek', 'Roscoe', 'Waffles', 'Jimbo', 'Maisey',
       'Howard', 'Lilly', 'Earl', 'Lola', 'Kevin', 'Yogi', 'Noah',
       'Bella', 'Grizzwald', 'Rusty', 'Gus', 'Stanley', 'Alfy', 'Koko',
       'Rey', 'Gary', 'Elliot', 'Louis', 'Jesse', 'Romeo', 'Bailey',
       'Duddles', 'Jack', 'Emmy', 'Steven', 'Beau', 'Snoopy', 'Shadow',
       'Terrance', 'Aja', 'Penny', 'Dante', 'Nelly', 'Ginger', 'Benedict',
       'Venti', 'Goose', 'Nugget', 'Cash', 'Coco', 'Jed', 'Sebastian',
       'Walter', 'Sierra', 'Monkey', 'Harry', 'Kody', 'Lassie', 'Rover',
       'Napolean', 'Dawn', 'Boomer', 'Cody', 'Rumble', 'Clifford',
       'Dewey', 'Scout', 'Gizmo', 'Cooper', 'Harold', 'Shikha', 'Jamesy',
       'Lili', 'Sammy', 'Meatball', 'Paisley', 'Albus', 'Neptune',
       'Quinn', 'Belle', 'Zooey', 'Dave', 'Jersey', 'Hobbes', 'Burt',
       'Lorenzo', 'Carl', 'Jordy', 'Milky', 'Trooper', 'Winston',
       'Sophie', 'Wyatt', 'Rosie', 'Thor', 'Oscar', 'Luna', 'Callie',
       'Cermet', 'George', 'Marlee', 'Arya', 'Einstein', 'Alice',
       'Rumpole', 'Benny', 'Aspen', 'Jarod', 'Wiggles', 'General',
       'Sailor', 'Astrid', 'Iggy', 'Snoop', 'Kyle', 'Leo', 'Riley',
       'Gidget', 'Noosh', 'Odin', 'Jerry', 'Charlie', 'Georgie', 'Rontu',
       'Cannon', 'Furzey', 'Daisy', 'Tuck', 'Barney', 'Vixen', 'Jarvis',
       'Mimosa', 'Pickles', 'Bungalo', 'Brady', 'Margo', 'Sadie', 'Hank',
       'Tycho', 'Stephan', 'Indie', 'Winnie', 'Bentley', 'Ken', 'Max',
       'Maddie', 'Pipsy', 'Monty', 'Sojourner', 'Odie', 'Arlo', 'Sunny',
       'Vincent', 'Lucy', 'Clark', 'Mookie', 'Meera', 'Buddy', 'Ava',
       'Rory', 'Eli', 'Ash', 'Tucker', 'Tobi', 'Chester', 'Wilson',
       'Sunshine', 'Lipton', 'Gabby', 'Bronte', 'Poppy', 'Rhino',
       'Willow', 'Orion', 'Eevee', 'Smiley', 'Logan', 'Moreton', 'Klein',
       'Miguel', 'Emanuel', 'Kuyu', 'Dutch', 'Pete', 'Scooter', 'Reggie',
       'Kyro', 'Samson', 'Loki', 'Mia', 'Malcolm', 'Dexter', 'Alfie',
       'Fiona', 'Mutt', 'Bear', 'Doobert', 'Beebop', 'Alexander',
       'Sailer', 'Brutus', 'Kona', 'Boots', 'Ralphie', 'Phil', 'Cupid',
       'Pawnd', 'Pilot', 'Ike', 'Mo', 'Toby', 'Sweet', 'Pablo', 'Nala',
       'Balto', 'Crawford', 'Gabe', 'Mattie', 'Jimison', 'Hercules',
       'Duchess', 'Harlso', 'Sampson', 'Sundance', 'Luca', 'Flash',
       'Finn', 'Peaches', 'Howie', 'Jazzy', 'Anna', 'Bo', 'Seamus',
       'Wafer', 'Chelsea', 'Tom', 'Moose', 'Florence', 'Autumn', 'Dido',
       'Eugene', 'Herschel', 'Strudel', 'Tebow', 'Chloe', 'Betty',
       'Timber', 'Binky', 'Dudley', 'Comet', 'Larry', 'Levi', 'Akumi',
       'Titan', 'Olivia', 'Alf', 'Oshie', 'Bruce', 'Chubbs', 'Sky',
       'Atlas', 'Eleanor', 'Layla', 'Rocky', 'Baron', 'Tyr', 'Bauer',
       'Swagger', 'Brandi', 'Mary', 'Moe', 'Halo', 'Augie', 'Craig',
       'Sam', 'Hunter', 'Pavlov', 'Maximus', 'Wallace', 'Ito', 'Milo',
       'Burke', 'Ollie', 'Cali', 'Lennon', 'Major', 'Duke', 'Reginald',
       'Sansa', 'Shooter', 'Django', 'Diogi', 'Sonny', 'Philbert',
       'Marley', 'Severus', 'Ronnie', 'Anakin', 'Bones', 'Mauve', 'Chef',
       'Doc', 'Sobe', 'Longfellow', 'Mister', 'Iroh', 'Baloo', 'Stubert',
       'Paull', 'Tickles', 'Timison', 'Davey', 'Pancake', 'Tyrone',
       'Snicku', 'Ruby', 'Brody', 'Rizzy', 'Mack', 'Butter', 'Nimbus',
       'Laika', 'Dobby', 'Juno', 'Maude', 'Lily', 'Newt', 'Benji', 'Nida',
       'Robin', 'Monster', 'BeBe', 'Remus', 'Mabel', 'Misty', 'Happy',
       'Mosby', 'Maggie', 'Leela', 'Ralphy', 'Brownie', 'Meyer', 'Stella',
       'Frank', 'Tonks', 'Lincoln', 'Oakley', 'Dale', 'Rizzo', 'Arnie',
       'Pinot', 'Dallas', 'Hero', 'Frankie', 'Stormy', 'Mairi', 'Loomis',
       'Godi', 'Kenny', 'Deacon', 'Timmy', 'Harper', 'Chipson', 'Combo',
       'Dash', 'Bell', 'Hurley', 'Jay', 'Mya', 'Strider', 'Wesley',
       'Solomon', 'Huck', "O'Malley", 'Blue', 'Finley', 'Sprinkles',
       'Heinrich', 'Shakespeare', 'Fizz', 'Chip', 'Grey', 'Roosevelt',
       'Gromit', 'Willem', 'Dakota', 'Dixie', 'Al', 'Jackson', 'Carbon',
```

'DonDon', 'Kirby', 'Lou', 'Nollie', 'Chevy', 'Tito', 'Louie',
'Rupert', 'Rufus', 'Brudge', 'Shadoe', 'Colby', 'Angel', 'Brat',
'Tove', 'Aubie', 'Kota', 'Eve', 'Glenn', 'Shelby', 'Sephie',
'Bonaparte', 'Albert', 'Wishes', 'Rose', 'Theo', 'Rocco', 'Fido',
'Emma', 'Spencer', 'Lilli', 'Boston', 'Brandonald', 'Corey',
'Leonard', 'Chompsky', 'Beckham', 'Devón', 'Gert', 'Watson',
'Rubio', 'Keith', 'Dex', 'Carly', 'Ace', 'Tayzie', 'Grizzie',
'Fred', 'Gilbert', 'Zoe', 'Stewie', 'Calvin', 'Lilah', 'Spanky',
'Jameson', 'Piper', 'Atticus', 'Blu', 'Dietrich', 'Divine',
'Tripp', 'Cora', 'Huxley', 'Keurig', 'Bookstore', 'Linus', 'Abby',
'Shaggy', 'Shiloh', 'Gustav', 'Arlen', 'Percy', 'Lenox', 'Sugar',
'Harvey', 'Blanket', 'Geno', 'Stark', 'Beya', 'Kilo', 'Kayla',
'Maxaroni', 'Doug', 'Edmund', 'Aqua', 'Theodore', 'Chase', 'Rorie',
'Simba', 'Charles', 'Bayley', 'Axel', 'Storkson', 'Remy',
'Chadrick', 'Kellogg', 'Buckley', 'Livvie', 'Terry', 'Hermione',
'Ralpher', 'Aldrick', 'Rooney', 'Crystal', 'Ziva', 'Stefan',
'Pupcasso', 'Puff', 'Flurpson', 'Coleman', 'Enchilada', 'Raymond',
'Rueben', 'Cilantro', 'Karll', 'Sprout', 'Blitz', 'Bloop',
'Lillie', 'Fred-Rick', 'Ashleigh', 'Kreggory', 'Sarge', 'Luther',
'Ivar', 'Jangle', 'Schnitzel', 'Panda', 'Berkeley', 'Ralphé',
'Charleson', 'Clyde', 'Harnold', 'Sid', 'Pippa', 'Otis', 'Carper',
'Bowie', 'Alexanderson', 'Suki', 'Barclay', 'Skittle', 'Ebby',
'Flávio', 'Smokey', 'Link', 'Jennifur', 'Ozzy', 'Bluebert',
'Stephanus', 'Bubbles', 'Zeus', 'Bertson', 'Nico',
'Michelangelope', 'Siba', 'Calbert', 'Curtis', 'Travis', 'Thumas',
'Kanu', 'Lance', 'Opie', 'Kane', 'Olive', 'Chuckles', 'Staniel',
'Sora', 'Beemo', 'Gunner', 'Lacy', 'Tater', 'Olaf', 'Cecil',
'Vince', 'Karma', 'Billy', 'Walker', 'Rodney', 'Klevin', 'Malikai',
'Bobble', 'River', 'Jebberson', 'Remington', 'Farfle', 'Jiminus',
'Clarkus', 'Finnegus', 'Cupcake', 'Kathmandu', 'Ellie', 'Katie',
'Kara', 'Adele', 'Zara', 'Ambrose', 'Jimothy', 'Bode', 'Terrenth',
'Reese', 'Chesterson', 'Lucia', 'Bisquick', 'Ralphson', 'Socks',
'Rambo', 'Rudy', 'Fiji', 'Rilo', 'Bilbo', 'Coopson', 'Yoda',
'Millie', 'Chet', 'Crouton', 'Daniel', 'Kaia', 'Murphy', 'Dotsy',
'Eazy-E', 'Coops', 'Fillup', 'Miley', 'Charl', 'Reagan', 'Yukon',
'CeCe', 'Cuddles', 'Claude', 'Jessiga', 'Carter', 'Ole', 'Pherb',
'Blipson', 'Reptar', 'Trevith', 'Berb', 'Bob', 'Colin', 'Brian',
'Oliviér', 'Grady', 'Kobe', 'Freddery', 'Bodie', 'Dunkin', 'Wally',
'Tupawc', 'Amber', 'Edgar', 'Teddy', 'Kingsley', 'Brockly',
'Richie', 'Molly', 'Vinscent', 'Cedrick', 'Hazel', 'Lolo', 'Eriq',
'Phred', 'Oddie', 'Maxwell', 'Geoff', 'Covach', 'Durg', 'Fynn',
'Ricky', 'Herald', 'Lucky', 'Ferg', 'Trip', 'Clarence', 'Hamrick',
'Brad', 'Pubert', 'Frönq', 'Derby', 'Lizzie', 'Ember', 'Blakely',
'Opal', 'Marq', 'Kramer', 'Barry', 'Gordon', 'Baxter', 'Mona',
'Horace', 'Crimson', 'Birf', 'Hammond', 'Lorelei', 'Marty',
'Brooks', 'Petrick', 'Hubertson', 'Gerbald', 'Oreo', 'Bruiser',
'Perry', 'Bobby', 'Jeph', 'Obi', 'Tino', 'Kulet', 'Sweets', 'Lupe',
'Tiger', 'Jiminy', 'Griffin', 'Banjo', 'Brandy', 'Lulu', 'Darrel',
'Taco', 'Joey', 'Patrick', 'Kreg', 'Todo', 'Tess', 'Thea',
'Ulysses', 'Toffee', 'Apollo', 'Asher', 'Glacier', 'Chuck',
'Champ', 'Ozzie', 'Griswold', 'Cheesy', 'Moofasa', 'Hector',
'Goliath', 'Kawhi', 'Emmie', 'Penelope', 'Willie', 'Rinna',
'Sabertooth', 'Mike', 'William', 'Dwight', 'Evy', 'Rascal',
'Linda', 'Tug', 'Tango', 'Grizz', 'Jerome', 'Crumpet', 'Jessifer',
'Izzy', 'Ralph', 'Sandy', 'Humphrey', 'Tassy', 'Juckson', 'Chuq',
'Tyrus', 'Karl', 'Godzilla', 'Vinnie', 'Kenneth', 'Herm', 'Bert',
'Striker', 'Donny', 'Pepper', 'Bernie', 'Buddah', 'Lenny', 'Wylie',
'Arnold', 'Zuzu', 'Mollie', 'Laela', 'Tedders', 'Superpup',
'Rufio', 'Jeb', 'Rodman', 'Jonah', 'Chesney', 'Henry', 'Bobbay',
'Mitch', 'Kaiya', 'Acro', 'Aiden', 'Obie', 'Dot', 'Shnuggles',
'Kendall', 'Kip', 'Jeffri', 'Steve', 'Mac', 'Fletcher', 'Kenzie',
'Pumpkin', 'Schnozz', 'Gustaf', 'Cheryl', 'Ed', 'Leonidas',
'Norman', 'Caryl', 'Scott', 'Taz', 'Darby', 'Jackie', 'Jazz',
'Franq', 'Pippin', 'Rolf', 'Snickers', 'Ridley', 'Cal', 'Bradley',
'Bubba', 'Tuco', 'Patch', 'Mojo', 'Batdog', 'Dylan', 'Mark',
'Jacob', 'JD', 'Alejandro', 'Scruffers', 'Pip', 'Julius', 'Tanner',
'Sparky', 'Anthony', 'Holly', 'Jett', 'Amy', 'Sage', 'Andy',
'Mason', 'Trigger', 'Antony', 'Creg', 'Traviss', 'Gin', 'Jeffrie',
'Danny', 'Ester', 'Pluto', 'Bloo', 'Edd', 'Willy', 'Spork', 'Herb',
'Damon', 'Peanut', 'Nigel', 'Cherokee', 'Butters', 'Hemry',
'Sandra', 'Fabio', 'Randall', 'Liam', 'Tommy', 'Ben', 'Raphael',
'Julio', 'Andru', 'Alphred', 'Kloey', 'Shawwn', 'Skye', 'Kollin',
'Alfredo', 'Ronduh', 'Billl', 'Saydee', 'Dug', 'Sully', 'Kirk',
'Ralf', 'Clarq', 'Jaspers', 'Samsom', 'Pancho', 'Harrison', 'Chaz',
'Jeremy', 'Jaycob', 'Leroi', 'Lambeau', 'Ruffles', 'Amélie',
'Bobb', 'Banditt', 'Kevon', 'Winifred', 'Hanz', 'Berta', 'Churlie',
'Zeek', 'Timofy', 'Maks', 'Jomathan', 'Kallie', 'Marvin', 'Spark',
'Gòrdón', 'Chuk', 'Jo', 'DayZ', 'Guss', 'Jareld', 'Torque', 'Ron',
'Skittles', 'Alfonso', 'Cleopatricia', 'Erik', 'Stu', 'Tedrick',
'Filup', 'Kial', 'Klint', 'Naphaniel', 'Big', 'Dook', 'Hall',
'Philippe', 'Kohl', 'Biden', 'Fwed', 'Genevieve', 'Joshwa',
'Daryl', 'Bradlay', 'Clybe', 'Keet', 'Carll', 'Pepe', 'Jockson',
'Octaviath', 'Josep', 'Lugan', 'Johm', 'Christoper'], dtype=object)

问题描述五

定义

doggo floofer pupper puppo等 列数据缺失

- 查询text中是否存在对应的单词（doggo floofer pupper puppo）
- 存在则放入新的列nickname中

代码

```
# twitter_archive_enhanced[twitter_archive_enhanced['text'].str.find('puppo')!=-1]
# nickname
for i in range(len(twitter_archive_enhanced)):
    if twitter_archive_enhanced.loc[i, 'text'].find("puppo")!=-1:
        twitter_archive_enhanced.loc[i, 'nickname'] = "puppo"

    if twitter_archive_enhanced.loc[i, 'text'].find("doggo")!=-1:
        twitter_archive_enhanced.loc[i, 'nickname'] = "doggo"

    if twitter_archive_enhanced.loc[i, 'text'].find("floofer")!=-1:
        twitter_archive_enhanced.loc[i, 'nickname'] = "floofer"

    if twitter_archive_enhanced.loc[i, 'text'].find("pupper")!=-1:
        twitter_archive_enhanced.loc[i, 'nickname'] = "pupper"
```

检测

```
twitter_archive_enhanced['nickname'].head(10)
```

```
0      NaN
1      NaN
2      NaN
3      NaN
4      NaN
5      NaN
6      NaN
7      NaN
8      NaN
9    doggo
Name: nickname, dtype: object
```

问题描述六

定义

rating_denominator数据缺失

代码

```
temp = twitter_archive_enhanced['text'].str.extract(r'(\d+\.?\d*\/\d+)')
temp = temp[0].str.split("/")

for i in range(len(twitter_archive_enhanced)):
    twitter_archive_enhanced.loc[i, 'rating_numerator'] = temp[i][0]
    twitter_archive_enhanced.loc[i, 'rating_denominator'] = temp[i][1]
```

检测

```
twitter_archive_enhanced.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2278 entries, 0 to 2277
Data columns (total 18 columns):
tweet_id                      2278 non-null int64
in_reply_to_status_id         0 non-null float64
in_reply_to_user_id           0 non-null float64
timestamp                     2278 non-null datetime64[ns]
source                        2278 non-null object
text                          2278 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2274 non-null object
rating_numerator              2278 non-null object
rating_denominator            2278 non-null object
name                          1534 non-null object
doggo                         2278 non-null object
floofer                       2278 non-null object
pupper                        2278 non-null object
puppo                         2278 non-null object
nickname                      388 non-null object
dtypes: datetime64[ns](1), float64(4), int64(1), object(12)
memory usage: 320.4+ KB
```

问题描述七

定义

twitter_archive_enhanced补充每一条推特数据的转发数和点赞数

代码

```
'''
    补充twitter_archive_enhanced数据集中retweet_count列和favorite_count列
'''
rtweet_count_list = []
favorite_count_list = []
for i in range(len(twitter_archive_enhanced)):
    id = twitter_archive_enhanced.loc[i]['tweet_id']
    try:
        t1 = tweet_json[tweet_json['id'] == id]["retweet_count"]
        t2 = tweet_json[tweet_json['id'] == id]["favorite_count"]
        rtweet_count_list.append(t1.iloc[0])
        favorite_count_list.append(t2.iloc[0])
    except:
        rtweet_count_list.append(np.nan)
        favorite_count_list.append(np.nan)
# rtweet_count_list
# favorite_count_list
```

```
twitter_archive_enhanced['favorite_count'] = favorite_count_list
twitter_archive_enhanced['retweet_count'] = rtweet_count_list
twitter_archive_enhanced[['favorite_count', 'retweet_count']].head(2)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	favorite_count	retweet_count
0	39492.0	8842.0
1	33786.0	6480.0

检测

```
# 查询id=890729181411237888的推特数据是否相等
print(tweet_json[tweet_json['id'] == 890729181411237888]['favorite_count'])
print(twitter_archive_enhanced[twitter_archive_enhanced['tweet_id'] == 890729181411237888]['favorite_count'])
```

```
7    66596
Name: favorite_count, dtype: int64
7    66596.0
Name: favorite_count, dtype: float64
```

```
# 查询id=890729181411237888的推特数据是否相等
print(tweet_json[tweet_json['id'] == 890729181411237888]['retweet_count'])
print(twitter_archive_enhanced[twitter_archive_enhanced['tweet_id'] == 890729181411237888]['retweet_count'])
```

```
7    19548
Name: retweet_count, dtype: int64
7    19548.0
Name: retweet_count, dtype: float64
```

问题描述八

定义

favorite_count 列和retweet_count中有几个NaN

- 用均值填充

代码

```
favorite_count_mean = twitter_archive_enhanced['favorite_count'].mean()
retweet_count_mean = twitter_archive_enhanced['retweet_count'].mean()
```

```
twitter_archive_enhanced['favorite_count'].fillna(favorite_count_mean, inplace = True)
twitter_archive_enhanced['retweet_count'].fillna(retweet_count_mean, inplace = True)
```

检测

```
twitter_archive_enhanced.info()
'''
favorite_count              2278 non-null float64
retweet_count               2278 non-null float64
已从2274 - > 2278
'''
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2278 entries, 0 to 2277
Data columns (total 20 columns):
tweet_id                    2278 non-null int64
```

```
in_reply_to_status_id         0 non-null float64
in_reply_to_user_id           0 non-null float64
timestamp                     2278 non-null datetime64[ns]
source                        2278 non-null object
text                          2278 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2274 non-null object
rating_numerator              2278 non-null object
rating_denominator            2278 non-null object
name                          1534 non-null object
doggo                         2278 non-null object
floofer                       2278 non-null object
pupper                        2278 non-null object
puppo                         2278 non-null object
nickname                      388 non-null object
favorite_count                2278 non-null float64
retweet_count                 2278 non-null float64
dtypes: datetime64[ns](1), float64(6), int64(1), object(12)
memory usage: 356.0+ KB
```

```
'\nfavorite_count                2278 non-null float64\nretweet_count                2278 non-null float64\n已从2274 - > 2278\n'
```

问题描述九

定义

doggo floofer pupper puppo 可以合并成一个列

- 已经合并，直接删除doggo floofer pupper puppo列

代码

```
twitter_archive_enhanced = twitter_archive_enhanced.drop(columns=["doggo", "floofer", "pupper", "puppo"])
```

检测

```
twitter_archive_enhanced.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2278 entries, 0 to 2277
Data columns (total 16 columns):
tweet_id                      2278 non-null int64
in_reply_to_status_id         0 non-null float64
in_reply_to_user_id           0 non-null float64
timestamp                     2278 non-null datetime64[ns]
source                        2278 non-null object
text                          2278 non-null object
retweeted_status_id           181 non-null float64
retweeted_status_user_id      181 non-null float64
retweeted_status_timestamp    181 non-null object
expanded_urls                 2274 non-null object
rating_numerator              2278 non-null object
rating_denominator            2278 non-null object
name                          1534 non-null object
nickname                      388 non-null object
favorite_count                2278 non-null float64
retweet_count                 2278 non-null float64
dtypes: datetime64[ns](1), float64(6), int64(1), object(8)
memory usage: 284.8+ KB
```

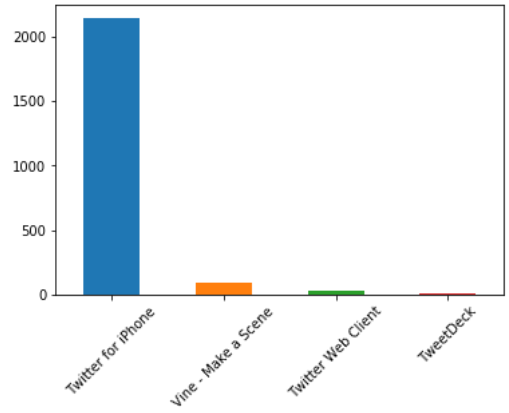
存储清理后的主数据集

```
twitter_archive_enhanced.to_csv("twitter_archive_master.csv.csv")
```

分析和可视化

```
# 分析或可视化代码
%matplotlib inline
twitter_data = pd.read_csv("twitter_archive_master.csv")
twitter_data['source'].value_counts().plot.bar(rot = 45)
```

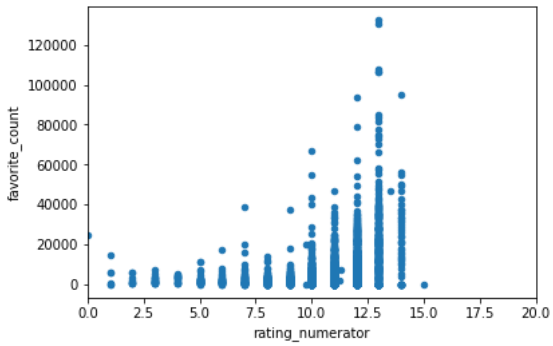
```
<matplotlib.axes._subplots.AxesSubplot at 0x23e0bd27748>
```



博主平时使用的是手机发送推特。

```
# 分析或可视化代码
twitter_data.plot.scatter(x="rating_numerator", y="favorite_count", xlim=(0,20))
```

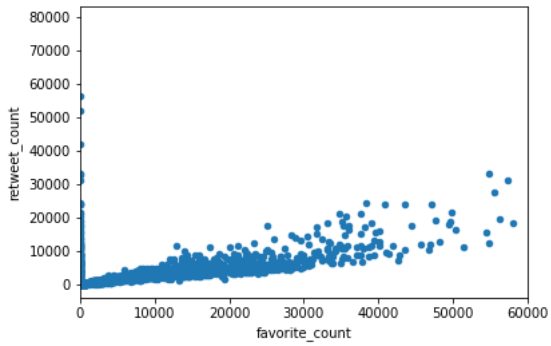
```
<matplotlib.axes._subplots.AxesSubplot at 0x23e0befe9e8>
```



分数和点赞数的关系图
从图中可以看出有时候在高分的时候，会获得搞点赞。

```
twitter_data[['favorite_count', "retweet_count"]].plot.scatter(x="favorite_count", y = "retweet_count", xlim=(0, 60000))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x23e0be786d8>
```



转发数和点赞数的关系图
点赞数和转发数成正相关性。

更多说明或总结等

提示：在完成 Notebook 的所有内容之后，还需要完成两篇文本和图片组成的 PDF 报告。因为这两篇报告中只是文字和图片，不需要包含代码，你可以使用文字编辑软件，比如 Word 来完成：

- 创建一个 300-600 字的书面报告，命名为 `wrangle_report.pdf`，在该报告中简要描述你的数据整理过程。这份报告可以看作是一份内部文档，供你的团队成员查看交流。
- 创建一个 250 字以上的书面报告，命名为 `act_report.pdf`，在该报告中，你可以与读者交流观点，展示你使用整理过的数据生成的可视化图表。这份报告可以看作是一份外部文档，如博客帖子或杂志文章。

提示：提交项目前建议删除 Notebook 中的所有提示性文字和注释，只保留自己的 Markdown 文本和代码注释。