**Wrangling report**

Data wrangling process.

**Stage (A) : Gather the information from 3 variant sources:**

▪ First source: It is a csv file on hand file called 'twitter-archive-enhanced.csv' , that was the most trivial part of the gathering process because all it needs is reading g via pandas data frame in a file called archive\_df.

▪ Second source: It is a tsv file was already saved in Udacity database and all we have to do is uploading this file programmatically and extracting Image-predictions.tsv from the URL provided. Then reading it via pandas Data frame in a file called image\_predictions\_df.

# reading Image Predictions File and extracting Image-predictions.tsv

folder\_name = 'image\_predictions\_file'

if not os.path.exists(folder\_name):

os.makedirs(folder\_name)

url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad\_image-predictions/image-predictions.tsv'

response = requests.get(url)

with open(os.path.join(folder\_name,

url.split('/')[-1]), mode = 'wb') as file:

file.write(response.content)

▪ Third source: I had to do the second choice which is using the file that already attached in the classroom and I studied the code related to this well. Anyway, I have done this and then read this file line by line to get the data , finally importing them into pandas data frame called api\_df. That was the most challenging part in the gathering process.

# reading (tweet-json.txt) file line by line

df\_list = []

with open('tweet-json.txt' ,'r') as file:

for line in file:

tweet = json.loads(line)

tweet\_id = tweet['id']

retweet\_count = tweet['retweet\_count']

fav\_count = tweet['favorite\_count']

user\_count = tweet['user']['followers\_count']

# Append to list of dictionaries.

df\_list.append({'tweet\_id': tweet\_id,

'retweet\_count': retweet\_count,

'favorite\_count': fav\_count,

'user\_count': user\_count})

**Reading files.**

# reading Enhanced Twitter Archive (.csv) and creating pandas dataframe.

archive\_df = pd.read\_csv('twitter-archive-enhanced.csv')

# reading Image-predictions (.tsv) and creating pandas dataframe.

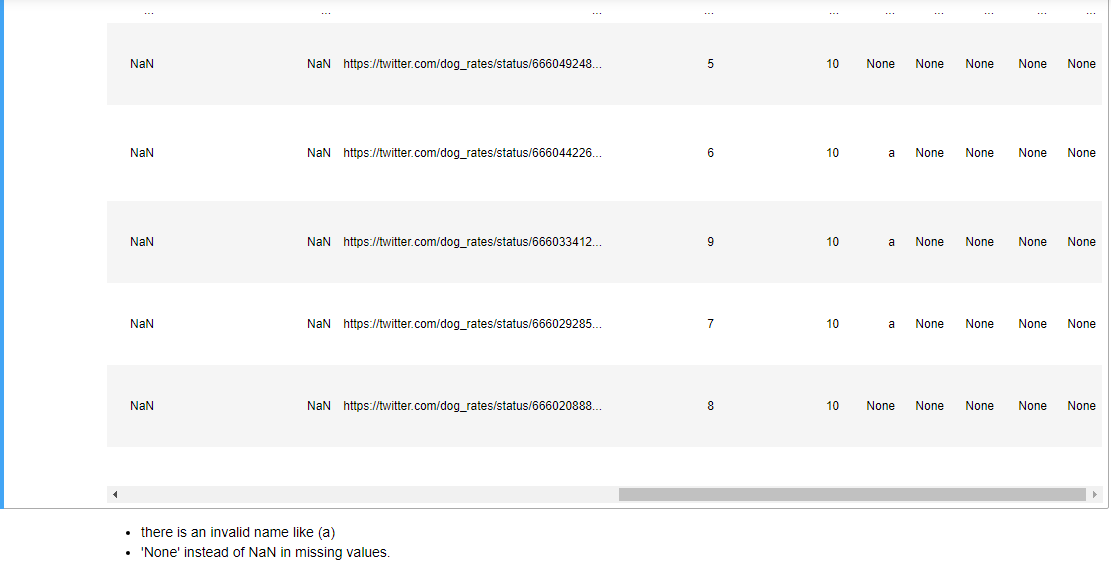
image\_predictions\_df = pd.read\_csv('image\_predictions\_file/image-predictions.tsv', sep='\t')

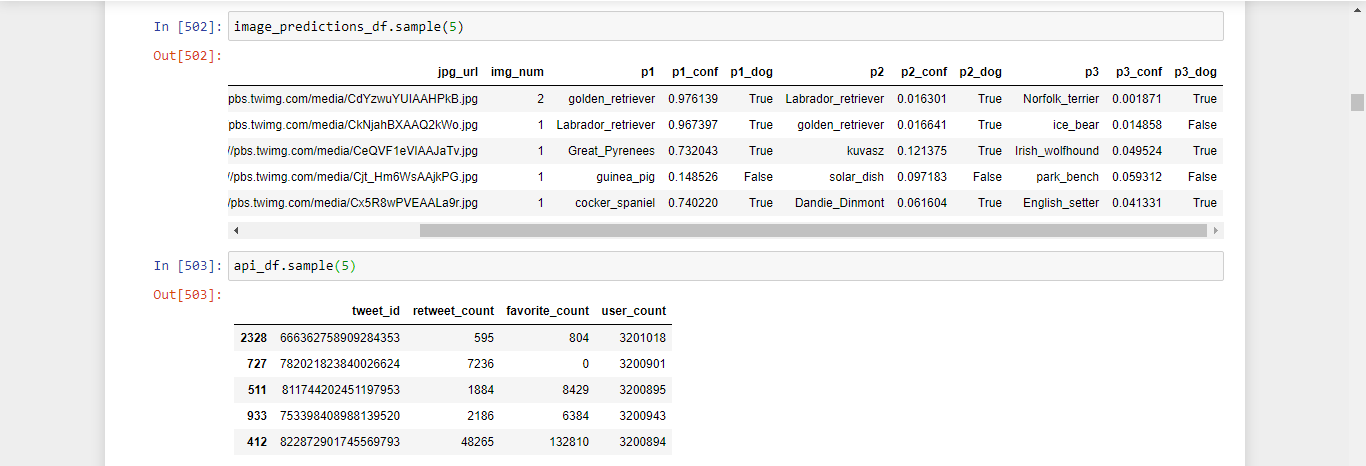
# # JSON objects to DataFrame:

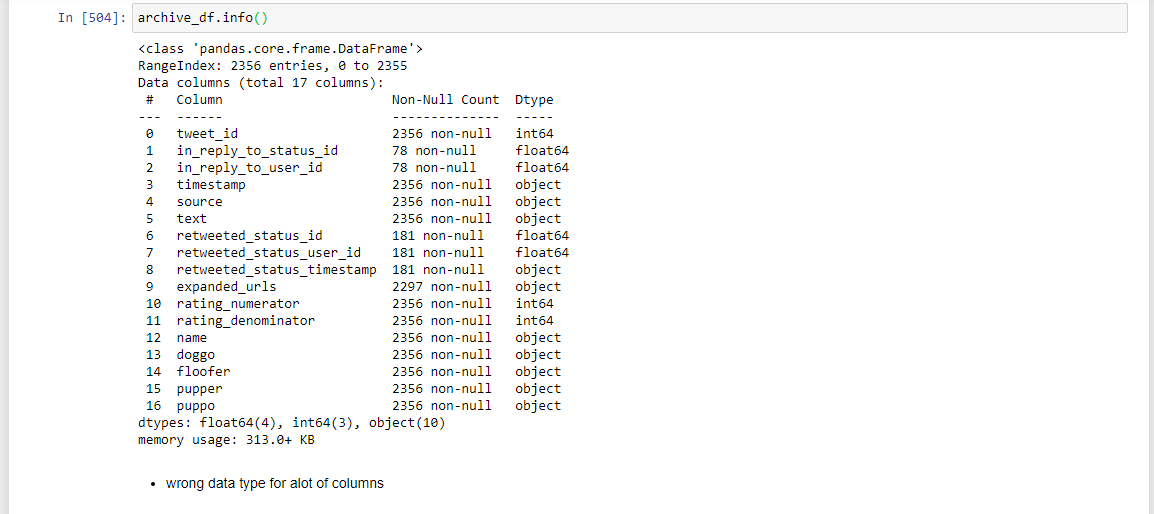
api\_df = pd.DataFrame(df\_list , columns = ['tweet\_id', 'retweet\_count', 'favorite\_count','user\_count'])

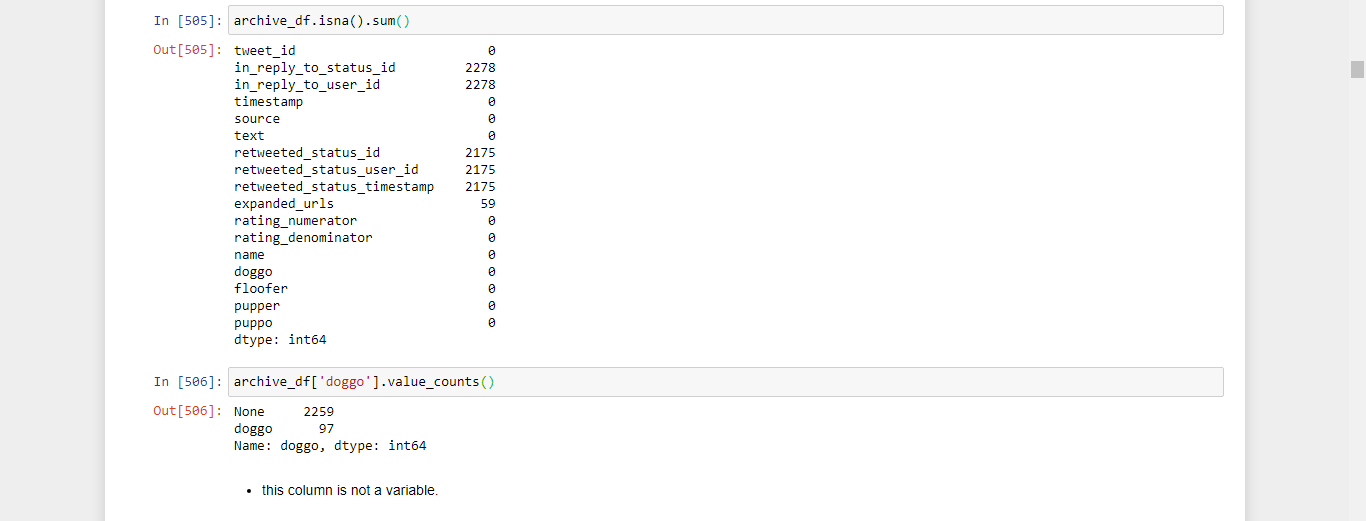
**Stage (B) : Assessing .**

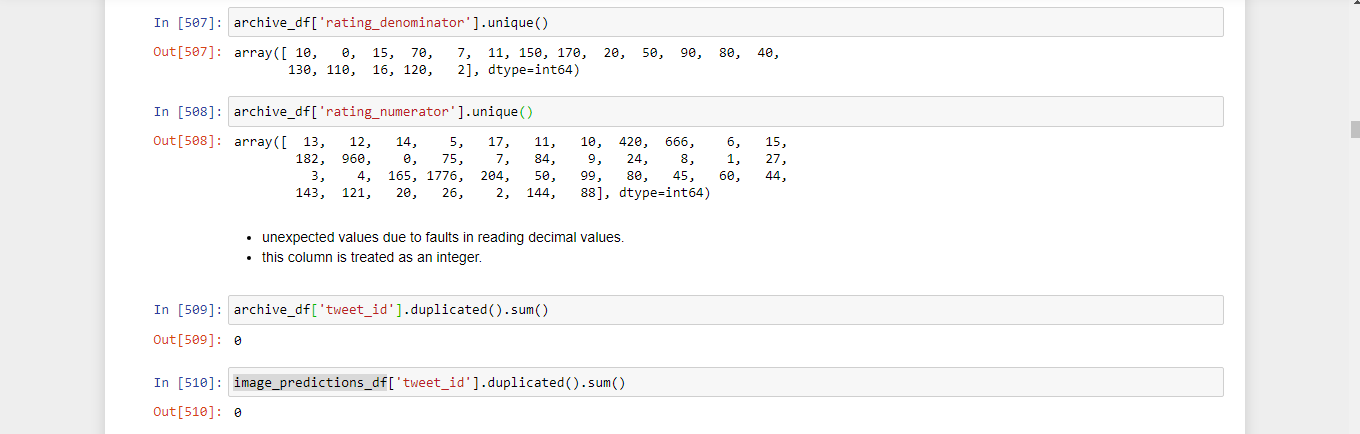
* **First, visually**

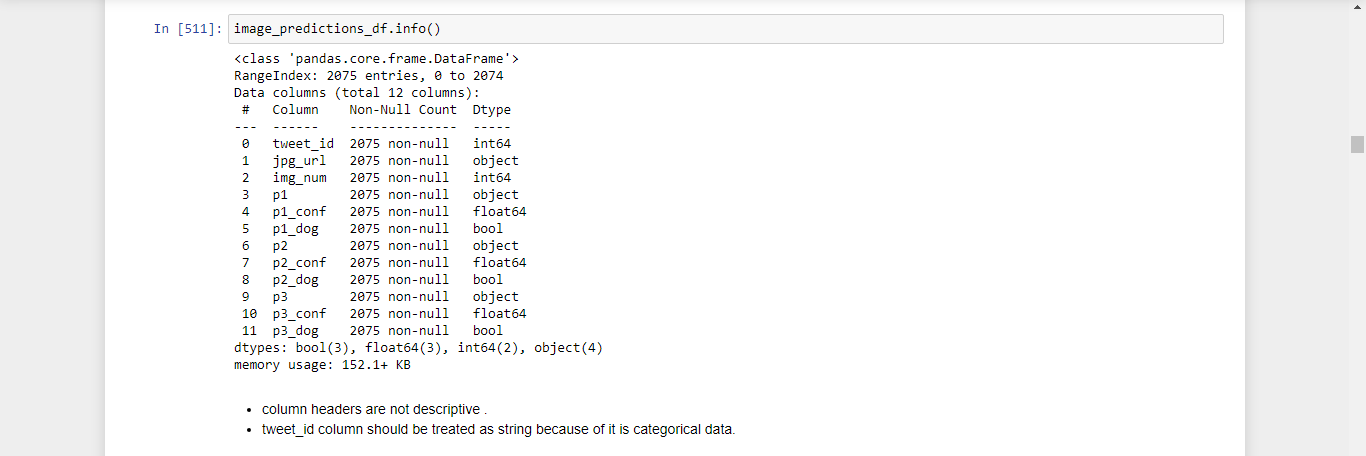
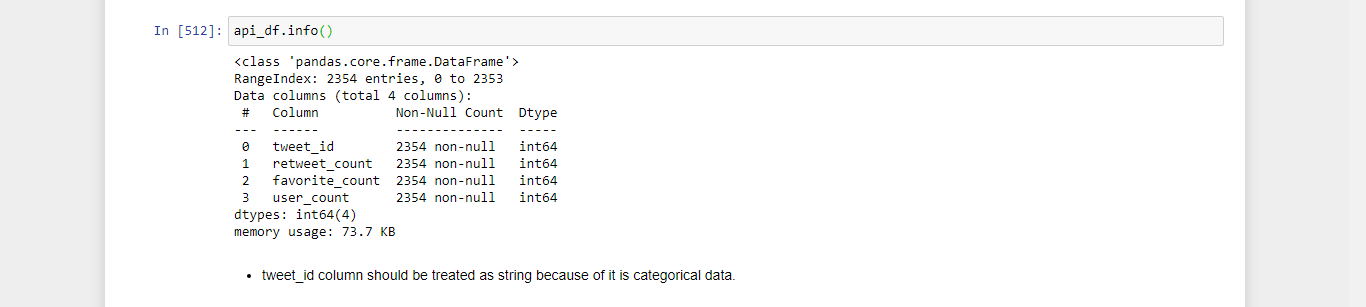
** archive\_df**

****

**Second: Programmatically**







After giving the data more investigation through two ways manually (display it on excel) and programmatically [ using info( ), describe( ), etc. ] and what have come in our summary was as following:

# Assessment summary.

## Quality issues:

##### *In archive\_df:*

* NaN is mistakenly written as **None** in: doggo,floofer,pupper,puppo, and name.
* tweet\_id type is **integer** instead of **str**.
* in\_reply\_to\_status\_id type is **float** instead of **str**.
* in\_reply\_to\_user\_id type is **float** instead of **str**.
* timestamp type is **str** instead of **DateTime**.
* source type is **str** instead of **category**.
* retweeted\_status\_id type is **float** instead of **str**.
* retweeted\_status\_user\_id type is **float** instead of **str**.
* retweeted\_status\_timestamp type is **str** instead of **DateTime**.
* name invalid names.
* rating\_numerator Ratings with decimal values incorrectly extracted
* rating\_numerator dtype is int instead of float.

##### *In image\_predictions\_df:*

* tweet\_id type is **integer** instead of **str**.
* columns headers are values not variable and not descriptive.

##### *Inapi\_df:*

* tweet\_id type is **integer** instead of **str**

## Tidiness Issues:

##### *In archive\_df:*

* doggo,floofer,pupper, and puppo should all be one column called e.g.dog\_stage.
* Some records are irrelevant (i.e. retweets or have replies).
* some recordes in archive\_df don't have images in image\_predictions\_df so I should filter the 3 datasets according to the records in image\_predictions\_df to get the original tweets with images.
* Data from the 3 datasets (archive\_df, image\_predictions\_df, and api\_df) can be combined in one DataFrame for simplicity.

**Stage (C) : Cleaning .**

### First, we need to take a copies for our datasets to avoid any bad thing for our original data frames that we get from Gathering step:

tweet\_clean = archive\_df.copy()

image\_clean = image\_predictions\_df.copy()

api\_clean = api\_df.copy()

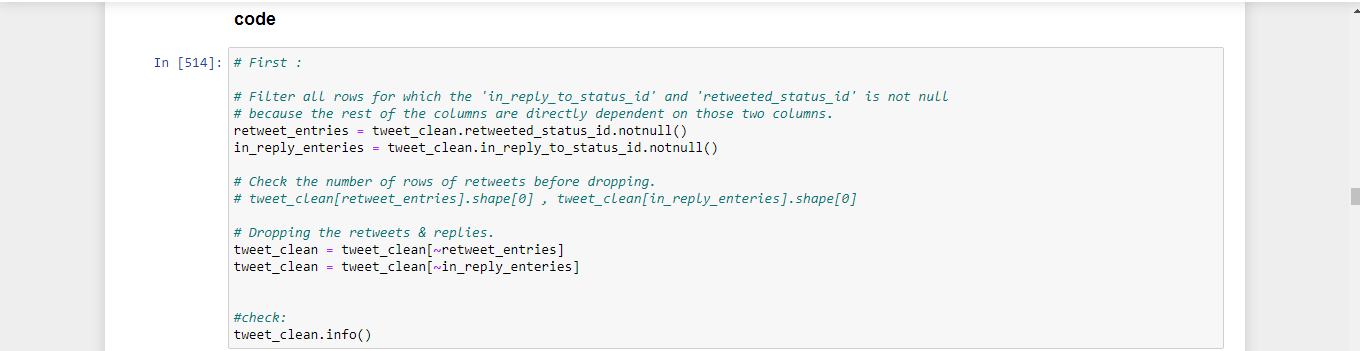
Second, we are going to have every single issue in Assessment Summary and solve it through applying [ Define , Code , Test ] strategy

## Define

##### *In `archive\_df:*

* Some records are irrelevant (i.e. retweets or have replies).
* some recordes in archive\_df don't have images in image\_predictions\_df so I should filter the 3 datasets according to the records in image\_predictions\_df to get the original tweets with images.

### Solution

* Drop every row that is retweeted\_status\_id.notnull()& in\_reply\_to\_status\_id.notnull()
* use the image\_predictions\_df to drop all irrelevant records from the other 2 datasets.
* through creating a list of tweet\_ids with images "tweets\_with\_image" and confirming itslength &use to get rid of tweets without images.

**Test**

tweet\_clean.shape , image\_clean.shape , api\_clean.shape

((1971, 17), (1971, 12), (1971, 4))

## Define

##### *In `archive\_df:*

* invalid names in name column.
* wrong representation for missing values as 'None'.

## solution:

* try extracting the right name from the text otherwise assign NaN to this value.
* use . replace() to convret them to NaNs.

## Test

tweet\_clean.name.value\_counts(dropna = False)

NaN 600

Charlie 11

Oliver 10

Cooper 10

Lucy 10

...

Ralphie 1

Pawnd 1

Harry 1

Goliath 1

Shelby 1

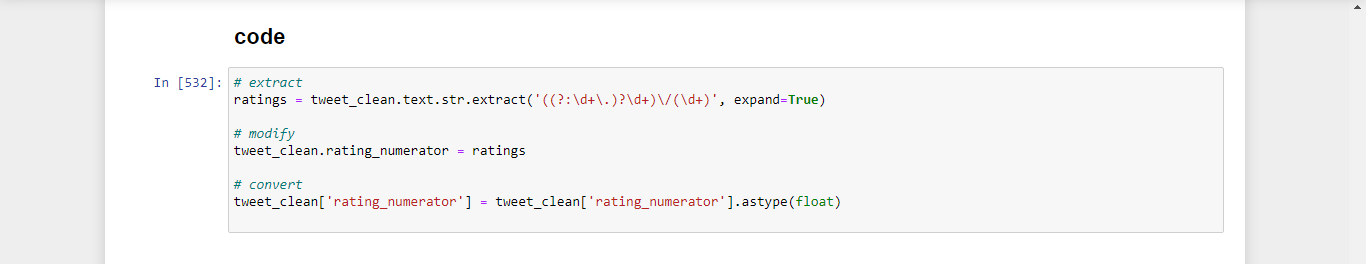
Name: name, Length: 931, dtype: int64

## Define

##### *In archive\_df:*

* Ratings with decimal values incorrectly extracted in rating\_numerator column
* rating\_numerator dtype is int instead of float.

## solution:

* extract the right decimal value using str.extract( ).
* convert rating\_numerator to float using astype().

## Test

tweet\_clean.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 1971 entries, 0 to 2355

Data columns (total 17 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 tweet\_id 1971 non-null int64

1 in\_reply\_to\_status\_id 0 non-null float64

2 in\_reply\_to\_user\_id 0 non-null float64

3 timestamp 1971 non-null object

4 source 1971 non-null object

5 text 1971 non-null object

6 retweeted\_status\_id 0 non-null float64

7 retweeted\_status\_user\_id 0 non-null float64

8 retweeted\_status\_timestamp 0 non-null object

9 expanded\_urls 1971 non-null object

10 rating\_numerator 1971 non-null float64

11 rating\_denominator 1971 non-null int64

12 name 1371 non-null object

13 doggo 1971 non-null object

14 floofer 1971 non-null object

15 pupper 1971 non-null object

16 puppo 1971 non-null object

dtypes: float64(5), int64(2), object(10)

memory usage: 357.2+ KB

tweet\_clean['rating\_numerator'].unique()

array([1.300e+01, 1.200e+01, 1.400e+01, 1.350e+01, 1.100e+01, 6.000e+00,

1.000e+01, 0.000e+00, 8.400e+01, 2.400e+01, 9.750e+00, 5.000e+00,

1.127e+01, 3.000e+00, 7.000e+00, 8.000e+00, 9.000e+00, 4.000e+00,

1.650e+02, 1.776e+03, 2.040e+02, 5.000e+01, 9.900e+01, 8.000e+01,

4.500e+01, 6.000e+01, 4.400e+01, 1.210e+02, 1.126e+01, 2.000e+00,

1.440e+02, 8.800e+01, 1.000e+00, 4.200e+02])

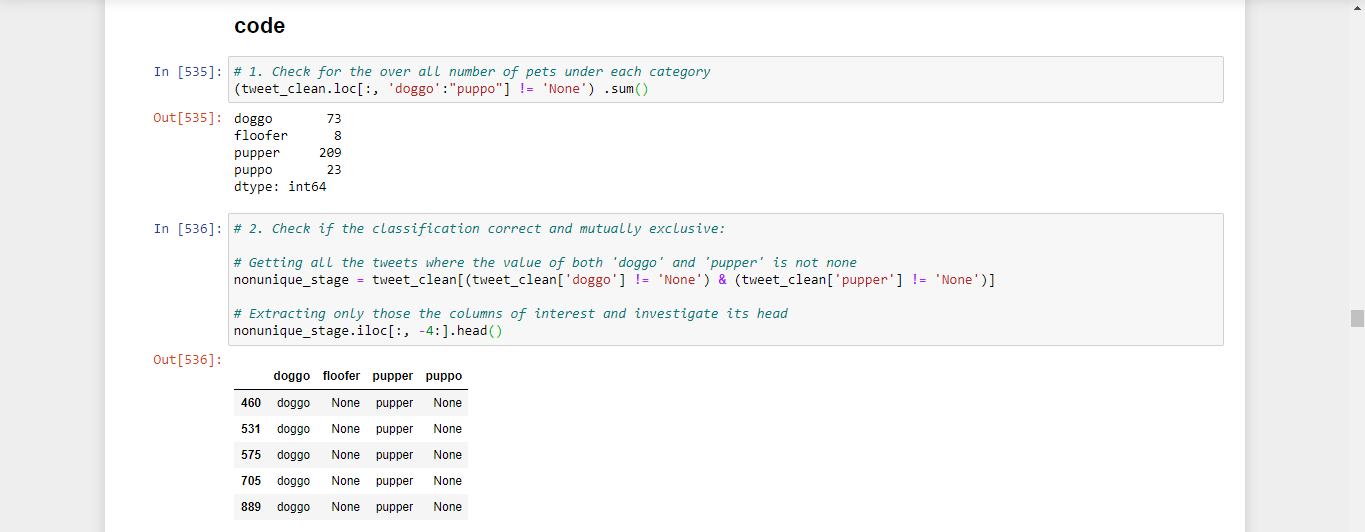
## Define

##### *In archive\_df:*

* NaN is mistakenly written as **None** in: doggo,floofer,pupper, and puppo.
* doggo,floofer,pupper, and puppo should all be one column called e.g.dog\_stage.

## Solution

* merge the last 4 columns to create new column called dog\_stage (further investegation needed)



## Test

tweet\_clean.dog\_stage.value\_counts(dropna=False)

NaN 1668

pupper 201

doggo 63

puppo 22

doggo-pupper 8

floofer 7

doggo-floofer 1

doggo-puppo 1

Name: dog\_stage, dtype: int64

## Define

##### *In archive\_df:*

* tweet\_id type is **integer** instead of **str**.
* in\_reply\_to\_status\_id type is **float** instead of **str**.
* in\_reply\_to\_user\_id type is **float** instead of **str**.
* timestamp type is **str** instead of **DateTime**.
* source type is **str** instead of **category**.
* retweeted\_status\_id type is **float** instead of **str**.
* retweeted\_status\_user\_id type is **float** instead of **str**.
* retweeted\_status\_timestamp type is **str** instead of **DateTime**.

##### *In image\_predictions\_df:*

* tweet\_id type is **integer** instead of **str**.

##### *In api\_df:*

* tweet\_id type is **integer** instead of **str**

#### **Solution:**

* Change values to str using .astype() method.
* Change values to category using .astype() method.
* Change values to datetime using pd.to\_datetime() method.

## Test

# test 1

assert tweet\_clean['tweet\_id'].dtype == 'O'

assert tweet\_clean['in\_reply\_to\_status\_id'].dtype == 'O'

assert tweet\_clean['in\_reply\_to\_user\_id'].dtype == 'O'

assert tweet\_clean['retweeted\_status\_id'].dtype == 'O'

assert tweet\_clean['retweeted\_status\_user\_id'].dtype == 'O'

assert image\_clean['tweet\_id'].dtype == 'O'

assert api\_clean['tweet\_id'].dtype == 'O'

assert tweet\_clean['source'].dtype == 'category'

# test 2

tweet\_clean.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 1971 entries, 0 to 2355

Data columns (total 14 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 tweet\_id 1971 non-null object

1 in\_reply\_to\_status\_id 1971 non-null object

2 in\_reply\_to\_user\_id 1971 non-null object

3 timestamp 1971 non-null datetime64[ns, UTC]

4 source 1971 non-null category

5 text 1971 non-null object

6 retweeted\_status\_id 1971 non-null object

7 retweeted\_status\_user\_id 1971 non-null object

8 retweeted\_status\_timestamp 0 non-null datetime64[ns]

9 expanded\_urls 1971 non-null object

10 rating\_numerator 1971 non-null float64

11 rating\_denominator 1971 non-null int64

12 name 1371 non-null object

13 dog\_stage 303 non-null object

dtypes: category(1), datetime64[ns, UTC](1), datetime64[ns](1), float64(1), int64(1), object(9)

memory usage: 297.6+ KB

# test 3

image\_clean.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 1971 entries, 0 to 2074

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 tweet\_id 1971 non-null object

1 jpg\_url 1971 non-null object

2 img\_num 1971 non-null int64

3 p1 1971 non-null object

4 p1\_conf 1971 non-null float64

5 p1\_dog 1971 non-null bool

6 p2 1971 non-null object

7 p2\_conf 1971 non-null float64

8 p2\_dog 1971 non-null bool

9 p3 1971 non-null object

10 p3\_conf 1971 non-null float64

11 p3\_dog 1971 non-null bool

dtypes: bool(3), float64(3), int64(1), object(5)

memory usage: 159.8+ KB

## Define

##### *In image\_predictions\_df:*

* columns headers are values not variable

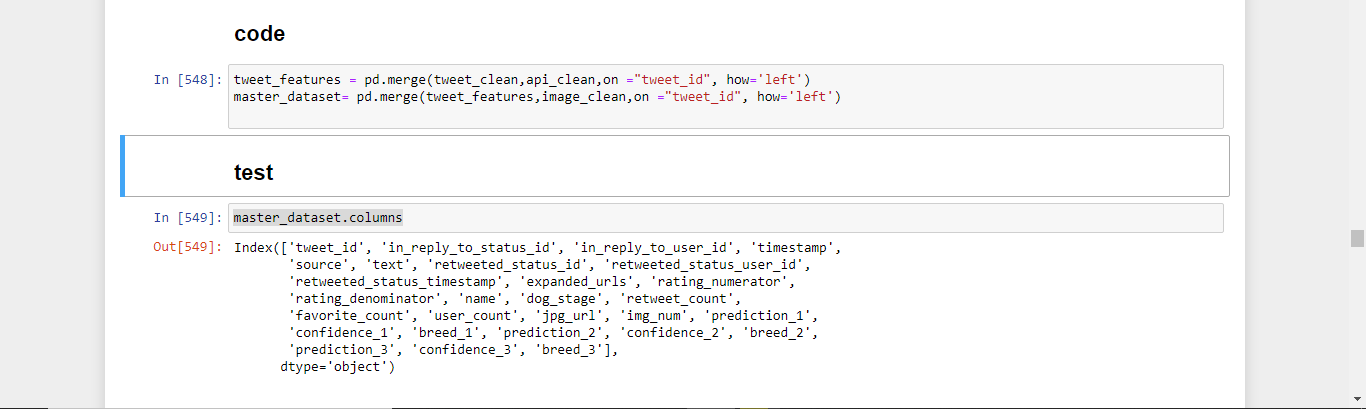
### solution:

* change columns headers using pd.wide\_to\_long()

## Define

* Data from the 3 datasets (archive\_df, image\_predictions\_df, and api\_df) can be combined in one DataFrame for simplicity.

## Solution

* while our 3 data sets have the same number of record we can use pd.merge()