The Wrangle Act of WeRateDogs

Table of Contents

In this project of data wrangling, we will go through three essential processes: gather, access and cleaning. There are three main datasets we are going to deal with, after gathering and accessing, we will then merge them into a master dataframe, and continue finishing the cleaning section.

Twitter Archive

The first data we will see is the twitter archive data, also the data we are going to work with the most. We will read it from a file called `twitter-archive-enhanced`, and name it as `df_archive`.

Take a brief look, we can see some pieces that we might need to handle later. To begin with, let's start from the data information:

```
<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
```

We found three issues in the information,

- The data type of datetime is a string.
- Some IDs are float.
- Missing values.

Next, take a look in the contents and we found:

source
<a href="http://twitter.com/download/iphone" r...
<a href="http://twitter.com/download/iphone" r...
<a href="http://twitter.com/download/iphone" r...
<a href="http://twitter.com/download/iphone" r...
<a href="http://twitter.com/download/iphone" r...

- Redundant contents in the source column which made them unreadable.

The information we need are squared in red, and the sources we sorted out from the dataframe are "Twitter for Iphone", "Make a Scene", "Twitter Web Client", and "TweetDeck".

Invalid ratings seen in numerator and denominator:

rating_numerator	rating_denominator
420	10
666	10
182	10
960	0

Strange names such as:

name
such
a
quite
quite
quite

The names and rating numbers came from texts and replies, thus some issues might occur in other columns.

Most important of all, we have to make the data as tidy as possible, which means some columns shouldn't be dispersed.

doggo	floofer	pupper	puppo
None	None	None	None
None	None	None	None
None	None	None	None
None	None	None	None

The columns above are all telling the same thing.

Image Prediction

In image prediction, we will gather the data from the Udacity server, and name it as `df_image_predictions`.

Since all the data we gathered were grouped by tweet ids, the issues weren't as much as the previous dataframe. By looking at the data information, minor issues we found are as follows:

```
<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
```

p1, p2, p3 in the data corresponded to the top three confident prediction of the dogs, in this case, they should be categories instead of strings.

The figure we are going to see next is the first count rows of same tweet pictures (url) with the different ids. Some pictures were seen more than one time, but with different ids, we can know that they are retweets.

```
https://pbs.twimg.com/media/ChK1tdBWwAQ1flD.jpg    2
https://pbs.twimg.com/media/CkjMx99UoAM2B1a.jpg    2
https://pbs.twimg.com/media/Ct2qO5PXEAE6eB0.jpg    2
https://pbs.twimg.com/media/CvT6IV6WEAQhhV5.jpg    2
https://pbs.twimg.com/media/CtVAvX-WIAAcGTf.jpg    2
Name: jpg_url, dtype: int64
```

Twitter API Query

The last dataframe we will generate is from the twitter api of our developer account. We will need four passwords: "consumer key", "consumer secret key", "access token", and "access secret token". We will then create our last dataframe with four columns which we need it later: "date_time", "tweet_id", "retweets", "favorites". Fortunately, the datafram we got this time is clean.

Cleaning Data

Issue Types

> **Quality Issues**

df_archive

- The data type of the timestamps should be datetime.
- IDs should be strings or integers.
- Missing values found in several columns.
- The source column should be more readable.
- Some rating numerators and denominators are not real ratings.
- Some names are detected as other words.

df_image_predictions

- p1, p2, p3 should be categorical instead of strings.
- Some pictures are retweets.

> **Tidiness Issues**

- The dog stage spread across 4 columns - doggo, floofer, pupper, puppo.
- Information about tweets is spread across three different dataframes.

Cleaning Data

Our process of data cleaning starts with tidiness issues. We will first make a copy of all three dataframes and merge them into a master dataframe to make our next cleaning steps more convenient. Secondly, we will combine all dog stages into one unique column, and check if every stages match the texts.

Tidiness Issues

1. Merging the 3 dataframes into one named "df_new" by `pd.merge()`

Result: The new dataframe have 31 columns in total.

1. Combining different dog stages into one column.

> - Count the number of all stages and thier combination using groupby.
> - Merge the stage columns and create a new column "stage".
> - Inspect the rows not within the four stages and refine them.

	doggo	floofer	pupper	puppo	count
0	None	None	None	None	1740
1	None	None	None	puppo	23
2	None	None	pupper	None	210
3	None	floofer	None	None	7
4	doggo	None	None	None	65
5	doggo	None	None	puppo	1
6	doggo	None	pupper	None	11
7	doggo	floofer	None	None	1

We can see there were some rows with multiple stages.

Now let's combine all columns with the '+' operator, and use comma to seperate columns with multiple stages. Next, we will look through every texts or pictures to confirm the correct dog stage. Finally, we will drop the old columns 'doggo', 'floofer', 'pupper', and 'puppo'. The cleaning result will be:

Result:

```
stage                         2058 non-null object
dtypes: bool(3), float64(7), int64(6), object(12)
memory usage: 424.1+ KB
```

Quality Issues

1. Drop NaNs, duplicates and unmeaningful columns.

Result:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1986 entries, 0 to 2057
Data columns (total 22 columns):
tweet_id              1986 non-null int64
timestamp             1986 non-null object
source                1986 non-null object
text                  1986 non-null object
expanded_urls         1986 non-null object
rating_numerator      1986 non-null int64
rating_denominator    1986 non-null int64
name                  1986 non-null object
retweets              1986 non-null int64
favorites             1986 non-null int64
jpg_url               1986 non-null object
img_num               1986 non-null int64
p1                    1986 non-null object
p1_conf               1986 non-null float64
p1_dog                1986 non-null bool
p2                    1986 non-null object
p2_conf               1986 non-null float64
p2_dog                1986 non-null bool
p3                    1986 non-null object
p3_conf               1986 non-null float64
p3_dog                1986 non-null bool
stage                 1986 non-null object
dtypes: bool(3), float64(3), int64(6), object(10)
memory usage: 316.1+ KB
```

1. Make the source content more readable using `apply()`.

Result:

source
Twitter for iPhone
Twitter for iPhone
Twitter for iPhone
Twitter for iPhone
Twitter for iPhone

1. Refine columns in image prediction.

> - Create prediction and confidence list and store the true algo with its level of confidence.
> - Loop through the boolean columns and append the prediction and confidence columns into our lists.
> - Apply our loop results into new columns.
> - Drop the not required columns of image prediction information.

Result:

```
prediction            1986 non-null object
confidence            1986 non-null float64
dtypes: float64(1), int64(5), object(8)
memory usage: 232.7+ KB
```

1. Get the correct dog gender from text column by filtering common phrases and assigning a gender. Using 'None' if no gender can be inferred from text.

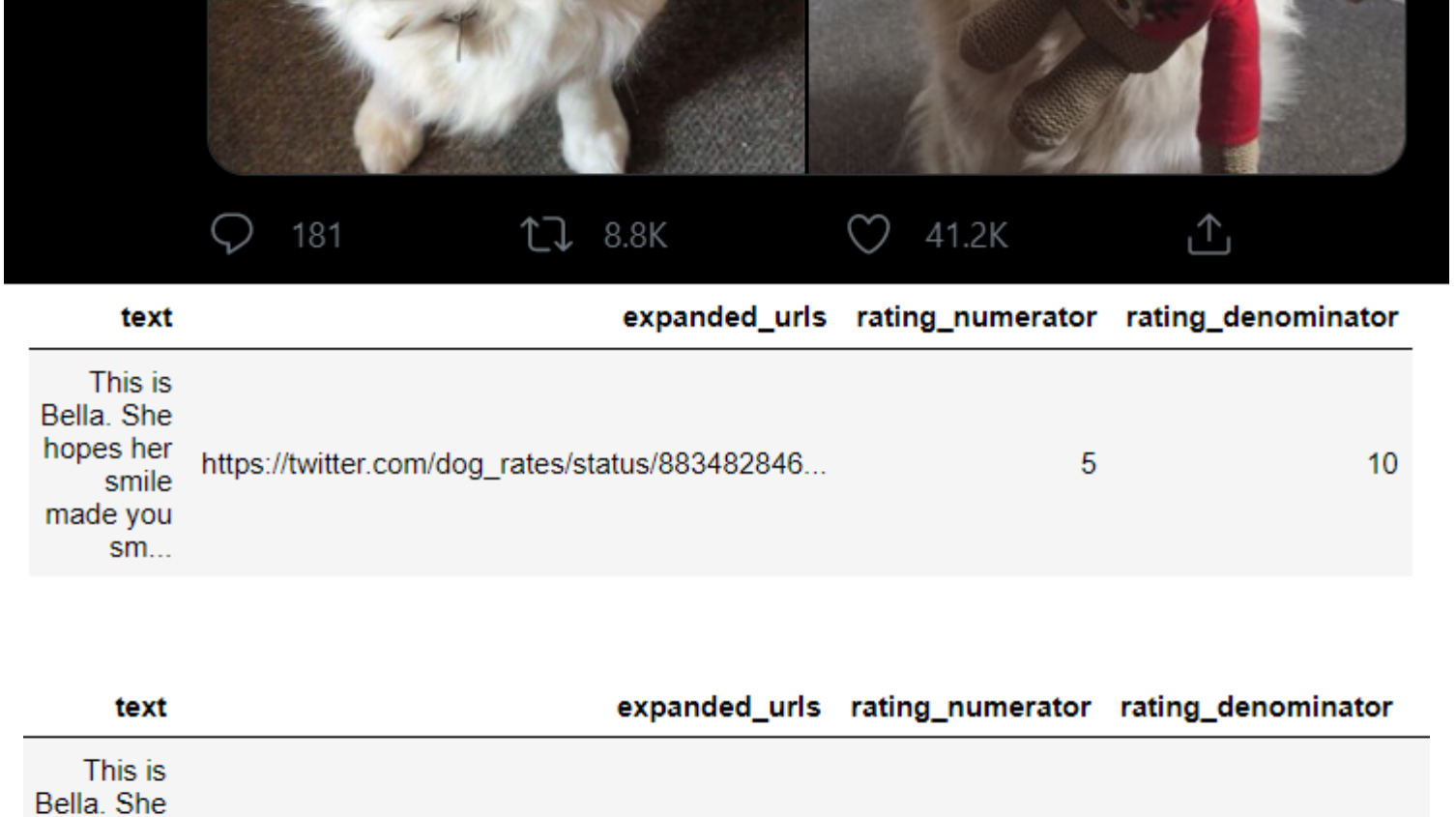
> - Create a male and female list and loop through all texts to check if it has one of pronouns of male or female.
> - Apply the results into a new column "gender".

Result:

```
None      1129
male       633
female     224
Name: gender, dtype: int64
```

1. Fix rating numerator and denominators that are not actually ratings by seeking all occurences where there are more than one #/# in 'text' column. Using `re.findall()` to scrap through the numerator column and `str.contains()` to view tweets with decimals in rating in 'text' column, then set the correct float ratings.

For example:



text	expanded_urls	rating_numerator	rating_denominator
This is Bella. She hopes her smile made you sm...	https://twitter.com/dog_rates/status/883482846...	5	10

Result:

text	expanded_urls	rating_numerator	rating_denominator
This is Bella. She hopes her smile made you sm...	https://twitter.com/dog_rates/status/883482846...	13.50	10.0

1. Remove unmeaningful names with lowercase by finding with `str.isupper()` and fill the nulls, then check the results with `str.islower()`.

Result: There are no rows with strange names.

1. Correct all datatypes and check them with `.dtypes`.

Result:

```
tweet_id                      object
timestamp             datetime64[ns]
source                      category
text                          object
expanded_urls                 object
rating_numerator             float64
rating_denominator           float64
name                          object
retweets                       int64
favorites                      int64
jpg_url                       object
stage                       category
prediction                    object
confidence                   float64
gender                      category
dtype: object
```

Finally, we will store all our progress above in a file called `twitter_archive_master.csv`, and use them in our data visualization.