wrangle_act

October 23, 2019

1 Gathering Data

```
In [0]: from google.colab import drive
        drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/cont
```

```
In [0]: import pandas as pd
        import requests
        import tweepy
        import json
        import sys
        import datetime as dt
        import matplotlib.pyplot as plt
        import re
        import numpy as np
```

```
In [0]: fn_twitter_archive_enhanced = '/content/drive/My Drive/Colab Notebooks/Data Wrangling/Pr
        fn_img_pred_local_file_name = '/content/drive/My Drive/Colab Notebooks/Data Wrangling/Pr
        fn_local_json_file = '/content/drive/My Drive/Colab Notebooks/Data Wrangling/Project/Dat
        fn_error_log = '/content/drive/My Drive/Colab Notebooks/Data Wrangling/Project/Data/json
        fn_local_json_file1 = '/content/drive/My Drive/Colab Notebooks/Data Wrangling/Project/Da
        fn_final_clean_file = '/content/drive/My Drive/Colab Notebooks/Data Wrangling/Project/Da

        img_pred_url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-

        consumer_key = 'MY CONSUMER KEY'
        consumer_secret = 'MY CONSUMER SECRET KEY'
        access_token = 'MY ACCESS TOKEN'
        access_secret = 'MY ACCESS SECRET'
```

```
In [0]: # Load the twitter enhance archive data
        df_twitter_archive = pd.read_csv(fn_twitter_archive_enhanced)
        df_twitter_archive.head()
```

```
Out[0]:              tweet_id  in_reply_to_status_id  ...  pupper puppo
        0  892420643555336193                    NaN  ...    None  None
```

```
1  892177421306343426                      NaN  ...    None  None
2  891815181378084864                      NaN  ...    None  None
3  891689557279858688                      NaN  ...    None  None
4  891327558926688256                      NaN  ...    None  None

[5 rows x 17 columns]
```

```
In [0]: # Download the image prediction data
        response = requests.get(img_pred_url)
```

```
In [0]: # Store the data in a local file
        with open(fn_img_pred_local_file_name, mode='wb') as file:
          file.write(response.content)
```

```
In [0]: # Load the data into a dataframe
        df_image_pred = pd.read_csv(fn_img_pred_local_file_name, sep='\t')
        df_image_pred.head()
```

```
Out[0]:              tweet_id  ... p3_dog
        0  666020888022790149  ...   True
        1  666029285002620928  ...   True
        2  666033412701032449  ...   True
        3  666044226329800704  ...   True
        4  666049248165822465  ...   True

        [5 rows x 12 columns]
```

1.0.1 Code for Downloading Twitter data. This did not work. I had Rate limit issues

```
In [0]: # Setup to access data from twitter
        auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
        auth.set_access_token(access_token, access_secret)

        api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)

        # For each tweet, get the tweet details
        logf = open(fn_error_log, 'w')
        outfile = open(fn_local_json_file, 'w')
        for index, row in df_twitter_archive.iterrows():
          t_id = row['tweet_id']
          try:
            data = api.get_status(t_id, tweet_mode='extended')._json + '\n'
            json.dump(data, outfile)
            logf.write('Success|' + str(t_id) + '\n')
            print(str(t_id))
          except:
            logf.write('Error|' + str(t_id) + '\n')

        logf.close()
        outfile.close()
        print('Done')
```

1.0.2 Alternate to twitter download. Used the file provided

```
In [0]: # Gather only those tweets which are not re-tweets
        # if line contains 'retweeted_status' key, it would mean its a retweet
        j_data = []
        with open(fn_local_json_file1, 'r') as j_file:
          for line in j_file:
            if 'retweeted_status' not in line:
              j_content = json.loads(line)
              j_data.append([j_content['id'], j_content['retweet_count'], j_content['favorite_co

        df_jdata = pd.DataFrame(j_data, columns=['tweet_id', 'retweet_count', 'favorite_count'])
        df_jdata.head()
```

```
Out[0]:              tweet_id  retweet_count  favorite_count
        0  892420643555336193           8853           39467
        1  892177421306343426           6514           33819
        2  891815181378084864           4328           25461
        3  891689557279858688           8964           42908
        4  891327558926688256           9774           41048
```

2 Assessing Data - eight (8) quality issues and two (2) tidiness issues

```
In [0]: # Create a copy of all the extracted/downloaded data
        df_twitter_arch_clean = df_twitter_archive.copy()
        df_image_pred_clean = df_image_pred.copy()
        df_jdata_clean = df_jdata.copy()
```

```
In [0]: df_twitter_arch_clean.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
```

```
In [0]: df_twitter_arch_clean.head(5)

Out[0]:              tweet_id  in_reply_to_status_id  ...  pupper puppo
        0  892420643555336193                    NaN  ...    None  None
        1  892177421306343426                    NaN  ...    None  None
        2  891815181378084864                    NaN  ...    None  None
        3  891689557279858688                    NaN  ...    None  None
        4  891327558926688256                    NaN  ...    None  None

        [5 rows x 17 columns]

In [0]: # check if any of these tweets are re-tweeted. These need to be removed
        df_twitter_arch_clean.retweeted_status_id.notnull().sum()

Out[0]: 181

In [0]: # On visual checks it was noticed that there were multiple tweets where the picture was
        # It had the text, 'we only rate dogs' lets check how many do we have like this
        str_search = 'WE ONLY RATE DOGS'
        df_twitter_arch_clean['text_uppercase'] = df_twitter_arch_clean.text.str.upper()
        df_twitter_arch_clean[df_twitter_arch_clean['text_uppercase'].str.contains(str_search)].

Out[0]: 63

In [0]: # Drop the additional column
        df_twitter_arch_clean.drop('text_uppercase', axis=1, inplace=True)

In [0]: # Rating numerator and denominator do not have null values
        # Are there any rows where the denominator is not 10
        df_temp = df_twitter_arch_clean[df_twitter_arch_clean['rating_denominator']!=10]
        df_temp.head()

Out[0]:                tweet_id  in_reply_to_status_id  ...  pupper puppo
        313  835246439529840640           8.352460e+17  ...    None  None
        342  832088576586297345           8.320875e+17  ...    None  None
        433  820690176645140481                    NaN  ...    None  None
        516  810984652412424192                    NaN  ...    None  None
        784  775096608509886464                    NaN  ...    None  None

        [5 rows x 17 columns]
```

```
In [0]: # Is this because the data is incorrect or due to the fact the data was extracted incorr
        df_temp[['tweet_id', 'text', 'rating_denominator', 'rating_numerator']].head(5)
```

```
Out[0]:                tweet_id  ... rating_numerator
        313  835246439529840640  ...              960
        342  832088576586297345  ...               11
        433  820690176645140481  ...               84
        516  810984652412424192  ...               24
        784  775096608509886464  ...                9

        [5 rows x 4 columns]
```

Visual checks show that the data is not being picked up corectly from the text field. The field rating_denominator is incorrectly picked up. Needs to be fixed. This will impact rating_numerator as well.

```
In [0]: # In case we are picking up the value from the text, lets check if there are any decimal
        df_twitter_arch_clean[df_twitter_arch_clean.text.str.contains(r"(\d+\.\d*\/\d+)")][['tex

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: UserWarning: This pattern has ma
  """Entry point for launching an IPython kernel.
```

```
Out[0]:                                                  text  ...  rating_denominator
        45    This is Bella. She hopes her smile made you sm...  ...                  10
        340   RT @dog_rates: This is Logan, the Chow who liv...  ...                  10
        695   This is Logan, the Chow who lived. He solemnly...  ...                  10
        763   This is Sophie. She's a Jubilant Bush Pupper. ...  ...                  10
        1689  I've been told there's a slight possibility he...  ...                  10
        1712  Here we have uncovered an entire battalion of ...  ...                  10

        [6 rows x 3 columns]
```

On visually checking there are decimal values. The numerator column should be a float type

```
In [0]: # Check for the values in rating_numerator as well
        df_twitter_arch_clean.rating_numerator.value_counts()
```

```
Out[0]: 12      558
        11      464
        10      461
        13      351
        9       158
        8       102
        7        55
        14       54
        5        37
        6        32
        3        19
```

```
4        17
1         9
2         9
420       2
0         2
15        2
75        2
80        1
20        1
24        1
26        1
44        1
50        1
60        1
165       1
84        1
88        1
144       1
182       1
143       1
666       1
960       1
1776      1
17        1
27        1
45        1
99        1
121       1
204       1
Name: rating_numerator, dtype: int64
```

There are values like 420, 0, 165 etc. lets check the actual text values for these tweets

```
In [0]: # Check for tweet texts where the numerator is 0 or greater than 20
        df_temp = df_twitter_arch_clean[(df_twitter_arch_clean['rating_numerator']==0) | (df_twi
        df_temp[['tweet_id', 'text', 'rating_numerator']].head(5)
```

```
Out[0]:                tweet_id  ... rating_numerator
        188  855862651834028034  ...              420
        189  855860136149123072  ...              666
        290  838150277551247360  ...              182
        313  835246439529840640  ...              960
        315  835152434251116546  ...                0

        [5 rows x 3 columns]
```

On checking these values visually as well, seems like these are valid values. No change required.

```
In [0]: # check for any duplicate rows in the dataframe
        df_twitter_arch_clean.duplicated().sum()

Out[0]: 0

In [0]: # check for the vales in the dog stage
        df_twitter_arch_clean.doggo.unique(), df_twitter_arch_clean.floofer.unique(), df_twitter

Out[0]: (array(['None', 'doggo'], dtype=object),
         array(['None', 'floofer'], dtype=object),
         array(['None', 'pupper'], dtype=object),
         array(['None', 'puppo'], dtype=object))

In [0]: # check to see if they are mutually unique
        df_twitter_arch_clean['combined_puppy_stage'] = (df_twitter_arch_clean.doggo + df_twitte
        df_twitter_arch_clean['combined_puppy_stage'] = df_twitter_arch_clean['combined_puppy_st
        df_twitter_arch_clean.combined_puppy_stage.value_counts()

Out[0]:                 1976
        pupper           245
        doggo             83
        puppo             29
        doggopupper       12
        floofer            9
        doggopuppo         1
        doggofloofer       1
        Name: combined_puppy_stage, dtype: int64

In [0]: # Check the dog name column
        df_twitter_arch_clean.name.value_counts()

Out[0]: None         745
        a             55
        Charlie       12
        Lucy          11
        Oliver        11
        Cooper        11
        Penny         10
        Lola          10
        Tucker        10
        Winston        9
        Bo             9
        Sadie          8
        the            8
        Toby           7
        Daisy          7
        Bailey         7
        an             7
        Buddy          7
```

```
Jax            6
Dave           6
Jack           6
Scout          6
Leo            6
Stanley        6
Rusty          6
Koda           6
Oscar          6
Bella          6
Milo           6
Chester        5
            ...
Jaycob         1
Biden          1
Vixen          1
Sandra         1
Jimbo          1
Oreo           1
Katie          1
Taz            1
Mo             1
Zuzu           1
Berkeley       1
space          1
Lillie         1
Venti          1
Zoe            1
Bayley         1
Keet           1
Willem         1
Ivar           1
Dietrich       1
Yukon          1
Patch          1
Laela          1
Wishes         1
Mutt           1
Callie         1
Tove           1
Kobe           1
Grady          1
Jebberson      1
Name: name, Length: 957, dtype: int64
```

Are names like all, None, an valid values - lets do a visual check. On visually checking, None - there does not seem to be a name in these tweets, seems okay. Should be set to Blank a, all, the, actually - again here it seems to be incorrectly extracted.

```
In [0]: # It seems that the dog names starting with a lower case are incorrect. Lets see some of
        df_twitter_arch_clean[df_twitter_arch_clean['name'].str[0].str.islower()]['name'].value_

Out[0]: a                55
        the               8
        an                7
        very              5
        quite             4
        one               4
        just              4
        not               2
        actually          2
        mad               2
        getting           2
        his               1
        infuriating       1
        unacceptable      1
        officially        1
        space             1
        old               1
        incredibly        1
        life              1
        light             1
        my                1
        such              1
        by                1
        all               1
        this              1
        Name: name, dtype: int64

In [0]: # Lets check the Image prediction dataframe
        df_image_pred_clean.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
```

```
In [0]: #check for any duplicates, since there are absolutely no null entries
        df_image_pred_clean.duplicated().sum(), df_image_pred_clean.tweet_id.duplicated().sum()

Out[0]: (0, 0)

In [0]: df_image_pred_clean.head()

Out[0]:              tweet_id  ... p3_dog
        0  666020888022790149  ...   True
        1  666029285002620928  ...   True
        2  666033412701032449  ...   True
        3  666044226329800704  ...   True
        4  666049248165822465  ...   True

        [5 rows x 12 columns]

In [0]: df_image_pred_clean.p1.value_counts()

Out[0]: golden_retriever             150
        Labrador_retriever           100
        Pembroke                      89
        Chihuahua                     83
        pug                           57
        chow                          44
        Samoyed                       43
        toy_poodle                    39
        Pomeranian                    38
        cocker_spaniel                30
        malamute                      30
        French_bulldog                26
        miniature_pinscher            23
        Chesapeake_Bay_retriever      23
        seat_belt                     22
        Siberian_husky                20
        German_shepherd               20
        Staffordshire_bullterrier     20
        Cardigan                      19
        web_site                      19
        teddy                         18
        Maltese_dog                   18
        Shetland_sheepdog             18
        Eskimo_dog                    18
        beagle                        18
        Lakeland_terrier              17
        Rottweiler                    17
```

```
Shih-Tzu                     17
Italian_greyhound            16
kuvasz                       16
                             ..
bow                           1
basketball                    1
teapot                        1
water_buffalo                 1
hay                           1
cuirass                       1
fountain                      1
ping-pong_ball                1
ice_lolly                     1
grille                        1
military_uniform              1
agama                         1
desktop_computer              1
African_hunting_dog           1
convertible                   1
piggy_bank                    1
king_penguin                  1
bonnet                        1
dining_table                  1
nail                          1
hare                          1
pitcher                       1
carousel                      1
maillot                       1
espresso                      1
candle                        1
rain_barrel                   1
ibex                          1
Scotch_terrier                1
trombone                      1
Name: p1, Length: 378, dtype: int64
```

```
In [0]: df_image_pred_clean.tail()
```

```
Out[0]:                 tweet_id  ... p3_dog
        2070  891327558926688256  ...   True
        2071  891689557279858688  ...  False
        2072  891815181378084864  ...   True
        2073  892177421306343426  ...   True
        2074  892420643555336193  ...  False

        [5 rows x 12 columns]
```

```
In [0]: # Json data file
        df_jdata_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2175 entries, 0 to 2174
Data columns (total 3 columns):
tweet_id          2175 non-null int64
retweet_count     2175 non-null int64
favorite_count    2175 non-null int64
dtypes: int64(3)
memory usage: 51.1 KB
```

3 Data Quality and Tidiness Issues

- DataFrame for Archived Twitter data

  1) Quality Issue - Base on the column retweeted_status_id, we noticed that there are around 181 retweets in this dataset. This should be removed.
  2) Quality Issue - There were tweets with the Text 'We only rate dogs'. These tweets should be excluded from our analysis.
  3) Quality Issue - There are rows where the rating_denominator is not 10. On checking visually for these rows, it seems that they were not parsed correctly from the text. Re-extract the Rating denominator from the text
  4) Quality Issue - For the rows where rating_denominator is incorrect, it was also noticed that the rating_numerator is incorrect. This needs to be correctly extracted.
  5) Tidiness Issue - Currently the rating_numerator and rating_denominator columns are type objects which would be string. We also noticed that some numerator values could be decimals. Change numerator to float and denominator to int
  6) Tidiness Issue - The puppy stage columns could have been consolidated to have a categorical value. However, there are 2 rows which have both values. For these 2, concatenate using ','.
  7) Quality Issue - The puppy stage columns have text like 'None' Can be fixed as Blank.
  8) Quality Issue - The timestamp column has additional details like +0000 which does not look necessary/correct.
  9) Tidiness Issue - The timestamp, retweeted_status_timestamp columns are object instead of datetime. Change timestamp to datetime type
  10) Tidiness Issue - in_reply_to_status_id, in_reply_to_user_id expanded_urls contains null values. These can be excluded from analysis.
  11) Quality Issue - The dog name column has 'None'. Should be Blank instead
  12) Quality Issue - Are names like all, None, an valid values. On visually checking, For values where it starts with a lowercase letter: a, all, the, actually - again here it seems to be incorrectly extracted. However on visually checking these records, it seems that there are no names in these tweets. Reset these values to Blank

- DataFrame for Image Prediction data

13) Quality Issue - There are 2356 entries in the Twitter Archived dataframe where as here there are only 2075 entries. We are missing approximately around 300 records at the least not considering any mismatches. Since these are missing, they cannot be merged back at the moment.
14) Tidiness Issue - The predicted p1, p2 and p3 have values which are not dogs. Relevant data would be to pick the prediction which is actually a dog with the highest probability. Create a scoring system to ascertain the best probaility of a dog

- DataFrame for Json data

15) This is the filtered list of tweets which are not re-tweets. This will be the source list for the rest of the data when combined together

16) Tidiness Issue - Pull in the data from Image Prediction and Json data to the twitter archive dataframe for analysis

4 Cleaning the data

```
In [0]: # DEFINE: Issue 1: Quality Issue - Base on the column retweeted_status_id, we noticed th
        # CLEAN: Issue 1:
        df_twitter_arch_clean = df_twitter_arch_clean[df_twitter_arch_clean.retweeted_status_id.
```

```
In [0]: # TEST CLEANED: Issue 1
        df_twitter_arch_clean.retweeted_status_id.notnull().sum()
```

```
Out[0]: 0
```

```
In [0]: # DEFINE: Issue 2: Quality Issue - On visual checks it was noticed that there were multi
        #                    It had the text, 'we only rate dogs'. Remove these rows
        # CLEAN: Issue 2:
        str_search = 'WE ONLY RATE DOGS'
        df_twitter_arch_clean['text_uppercase'] = df_twitter_arch_clean.text.str.upper()
        df_twitter_arch_clean = df_twitter_arch_clean[~df_twitter_arch_clean['text_uppercase'].s
```

```
In [0]: # TEST CLEANED: Issue 2
        df_twitter_arch_clean[df_twitter_arch_clean['text_uppercase'].str.contains(str_search)].
```

```
Out[0]: 0
```

```
In [0]: # DEFINE:

        # Issue 3: Quality Issue - There are rows where the rating_denominator is not 10. On che
        #           Re-extract the Rating denominator from the text

        # Issue 4: Quality Issue - For the rows where rating_denominator is incorrect, it was al

        # Issue 5: Quality/Tidiness Issue - Currently the rating_numerator and rating_denominato
        #           We also noticed that some numerator values could be decimals. Add clean colu
# CLEAN: Issue 3, 4, 5:
# For each row, get the twitter text, Split by pattern %d/%d, take the one which has 10
df_twitter_arch_clean['numerator_clean'] = 0.0
df_twitter_arch_clean['denominator_clean'] = 0.0
df_twitter_arch_clean.numerator_clean.astype(float)
df_twitter_arch_clean.denominator_clean.astype(float)

for index, row in df_twitter_arch_clean.iterrows():
  s_split = re.findall('\d+\.*\d*/\d+', row.text) # this pattern should give us both dec
  for s in s_split:
    num = float(s.split('/')[0])
    denom = float(s.split('/')[1])
    if(denom==10):
      df_twitter_arch_clean.loc[index,'numerator_clean'] = num
      df_twitter_arch_clean.loc[index,'denominator_clean'] = denom
      break
```

```
In [0]: # TEST CLEANED:
        # Issue 3 and 4
        df_twitter_arch_clean[df_twitter_arch_clean['rating_denominator']!=10][['tweet_id', 'tex
```

```
Out[0]:                tweet_id  ... denominator_clean
        313  835246439529840640  ...              10.0
        342  832088576586297345  ...               0.0
        433  820690176645140481  ...               0.0
        516  810984652412424192  ...               0.0
        902  758467244762497024  ...               0.0

        [5 rows x 6 columns]
```

```
In [0]: # Check why there are 0 value denominators
        df_twitter_arch_clean[df_twitter_arch_clean.denominator_clean==0][['tweet_id', 'text']]
```

```
Out[0]:                 tweet_id                                               text
        342   832088576586297345           @docmisterio account started on 11/15/15
        433   820690176645140481  The floofs have been released I repeat the flo...
        516   810984652412424192  Meet Sam. She smiles 24/7 &amp; secretly aspir...
        902   758467244762497024  Why does this never happen at my front door...
        1120  731156023742988288  Say hello to this unbelievably well behaved sq...
        1228  713900603437621249  Happy Saturday here's 9 puppers on a bench. 99...
        1254  710658690886586372  Here's a brigade of puppers. All look very pre...
        1274  709198395643068416  From left to right:\nCletus, Jerome, Alejandro...
        1351  704054845121142784  Here is a whole flock of puppers.  60/50 I'll ...
        1433  697463031882764288  Happy Wednesday here's a bucket of pups. 44/40...
        1598  686035780142297088  Yes I do realize a rating of 4/20 would've bee...
        1634  684225744407494656  Two sneaky puppers were not initially seen, mo...
        1635  684222868335505415  Someone help the girl is being mugged. Several...
        1663  682808988178739200  I'm aware that I could've said 20/16, but here...
```

```
1779  677716515794329600  IT'S PUPPERGEDDON. Total of 144/120 ...I think...
1843  675853064436391936  Here we have an entire platoon of puppers. Tot...
```

For the ones that are blank in our extract, on doing a visual check, we noticed some them indeed had valid values. However, some of them did not. For eg. 204/170 is a valid rating. However, text 'account started on 11/15/15' does not give us the right information. Some of them had 50/50 and 13/10 in the same text and the other had 45/50. There was no pattern to distinguish beteween this. For now, we are considering only ratings with a denominator of 10

```
In [0]: # TEST CLEANED:
        # Issue 5 - Check the column types added
        df_twitter_arch_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2121 entries, 0 to 2355
Data columns (total 21 columns):
tweet_id                      2121 non-null int64
in_reply_to_status_id         77 non-null float64
in_reply_to_user_id           77 non-null float64
timestamp                     2121 non-null object
source                        2121 non-null object
text                          2121 non-null object
retweeted_status_id           0 non-null float64
retweeted_status_user_id      0 non-null float64
retweeted_status_timestamp    0 non-null object
expanded_urls                 2063 non-null object
rating_numerator              2121 non-null int64
rating_denominator            2121 non-null int64
name                          2121 non-null object
doggo                         2121 non-null object
floofer                       2121 non-null object
pupper                        2121 non-null object
puppo                         2121 non-null object
combined_puppy_stage          2121 non-null object
text_uppercase                2121 non-null object
numerator_clean               2121 non-null float64
denominator_clean             2121 non-null float64
dtypes: float64(6), int64(3), object(12)
memory usage: 444.5+ KB
```

```
In [0]: #DEFINE

        # Issue 6: Tidiness Issue - The puppy stage columns could have been consolidated to have

        # Issue 7: Quality Issue - The puppy stage columns have text like 'None' Can be fixed as

        # CLEAN Issue 7:
        #The puppy stage columns have text like 'None' Can be fixed as Blank.
        df_twitter_arch_clean.doggo.replace('None', '', inplace=True)
        df_twitter_arch_clean.floofer.replace('None', '', inplace=True)
        df_twitter_arch_clean.pupper.replace('None', '', inplace=True)
        df_twitter_arch_clean.puppo.replace('None', '', inplace=True)
```

```
In [0]: # TEST Issue 7
        df_twitter_arch_clean.doggo.str.contains('None').sum(),df_twitter_arch_clean.floofer.str
```

```
Out[0]: (0, 0, 0, 0)
```

```
In [0]: # CLEAN Issue 6
        # The puppy stage columns could have been consolidated to have a categorical value. Howe
        df_twitter_arch_clean['combined_puppy_stage'] = ''
        for index, row in df_twitter_arch_clean.iterrows():
          doggo, floofer, pupper, puppo = row.doggo, row.floofer, row.pupper, row.puppo
          str_dogstage = ''
          if(doggo!=''):
            str_dogstage=doggo
          if(floofer!=''):
            if(str_dogstage==''):
              str_dogstage = floofer
            else:
              str_dogstage = str_dogstage + ',' + floofer
          if(pupper!=''):
            if(str_dogstage==''):
              str_dogstage = pupper
            else:
              str_dogstage = str_dogstage + ',' + pupper
          if(puppo!=''):
            if(str_dogstage==''):
              str_dogstage = puppo
            else:
              str_dogstage = str_dogstage + ',' + puppo

          df_twitter_arch_clean.loc[index,'combined_puppy_stage']=str_dogstage
```

```
In [0]: # TEST Issue 6
        # Check the value counts
        df_twitter_arch_clean['combined_puppy_stage'].value_counts()
```

```
Out[0]:                  1777
        pupper            224
        doggo              75
        puppo              24
        doggo,pupper       10
        floofer             9
        doggo,puppo         1
        doggo,floofer       1
        Name: combined_puppy_stage, dtype: int64
```

```
In [0]: # DEFINE
        # Issue 8 Quality Issue - The timestamp column has additional details like +0000 which d

        # Issue 9 Tidiness Issue - The timestamp, retweeted_status_timestamp columns are object

        #CLEAN Issues 8 and 9
        df_twitter_arch_clean['timestamp_clean'] = pd.to_datetime(df_twitter_arch_clean['timesta
        df_twitter_arch_clean['retweeted_status_timestamp_clean'] = pd.to_datetime(df_twitter_ar

In [0]: # TEST Issues 8 and 9
        print(str(df_twitter_arch_clean['timestamp'].dtype) + ' ' + str(df_twitter_arch_clean['t
        print(str(df_twitter_arch_clean['retweeted_status_timestamp'].dtype) + ' ' + str(df_twit

object datetime64[ns, UTC]
object datetime64[ns]


In [0]: df_twitter_arch_clean.timestamp_clean.head()

Out[0]: 0   2017-08-01 16:23:56+00:00
        1   2017-08-01 00:17:27+00:00
        2   2017-07-31 00:18:03+00:00
        3   2017-07-30 15:58:51+00:00
        4   2017-07-29 16:00:24+00:00
        Name: timestamp_clean, dtype: datetime64[ns, UTC]

In [0]: # DEFINE Issue 12 Quality Issue - Are names like all, None, an valid values. On visually
        # a, all, the, actually - again here it seems to be incorrectly extracted. However on vi
        # Reset these values to Blank

        # CLEAN Issue 12
        df_twitter_arch_clean['name_clean'] = df_twitter_arch_clean['name']
        for index, row in df_twitter_arch_clean.iterrows():
          str_name_clean = row['name_clean']
          if(str(str_name_clean)[0].islower()):
            df_twitter_arch_clean.loc[index,'name_clean']='None'

In [0]: # TEST Issue 12
        df_twitter_arch_clean[df_twitter_arch_clean['name_clean'].str[0].str.islower()]['name_cl

Out[0]: Series([], Name: name_clean, dtype: int64)

In [0]: df_twitter_arch_clean.name_clean.value_counts()

Out[0]: None         731
        Lucy          11
        Charlie       11
        Cooper        10
        Oliver        10
```

```
Tucker         9
Penny          9
Sadie          8
Winston        8
Lola           8
Daisy          7
Toby           7
Bella          6
Bo             6
Jax            6
Bailey         6
Koda           6
Stanley        6
Oscar          6
Bentley        5
Buddy          5
Louis          5
Rusty          5
Milo           5
Leo            5
Chester        5
Dave           5
Scout          5
Brody          4
Oakley         4
            ...
Rooney         1
Jed            1
Dotsy          1
Ester          1
Wishes         1
Patch          1
Raphael        1
Yukon          1
Kendall        1
Baloo          1
Alf            1
Nugget         1
Robin          1
Zuzu           1
Berkeley       1
Rizzy          1
Pipsy          1
Lillie         1
Venti          1
Zeek           1
Marvin         1
Zoe            1
```

```
Bayley         1
Iggy           1
Willem         1
Mo             1
Ivar           1
Leela          1
Sprinkles      1
Jebberson      1
Name: name_clean, Length: 931, dtype: int64
```

```
In [0]: #DEFINE Issue 11 Quality Issue - The dog name column has 'None'. Should be Blank instead
        # CLEAN Issue 11
        df_twitter_arch_clean.name_clean.replace(['None'], '', inplace=True)
```

```
In [0]: #TEST Issue 11
        df_twitter_arch_clean[df_twitter_arch_clean.name_clean=='None'].shape[0]
```

```
Out[0]: 0
```

```
In [0]: # DEFINE: 14 Tidiness Issue - The predicted p1, p2 and p3 have values which are not dogs
        # Relevant data would be to pick the prediction which is actually a dog with the highest
        # Create a scoring system to ascertain the best probaility of a dog

        # CLEAN: 14

        truefalsemap = {
            True: 1,
            False: 0
        }

        df_image_pred_clean['p1_score'] = df_image_pred_clean['p1_conf'] * df_image_pred_clean['
        df_image_pred_clean['p2_score'] = df_image_pred_clean['p2_conf'] * df_image_pred_clean['
        df_image_pred_clean['p3_score'] = df_image_pred_clean['p3_conf'] * df_image_pred_clean['
        df_image_pred_clean.head()
```

```
Out[0]:              tweet_id  ...  p3_score
        0  666020888022790149  ...  0.061428
        1  666029285002620928  ...  0.072010
        2  666033412701032449  ...  0.116197
        3  666044226329800704  ...  0.222752
        4  666049248165822465  ...  0.154629

        [5 rows x 15 columns]
```

```
In [0]: # CLEAN: 14 Contd.

        # Store the data in a separate column called predicted_dog
        # in case none of the prediction is a dog, keep it blank
        df_image_pred_clean['predicted_dog'] = ''
        for index, row in df_image_pred_clean.iterrows():
          if((row['p1_score']>row['p2_score']) & (row['p1_score']>row['p3_score'])):
            df_image_pred_clean.loc[index,'predicted_dog']=row['p1']
          elif((row['p2_score']>row['p1_score']) & (row['p2_score']>row['p3_score'])):
            df_image_pred_clean.loc[index,'predicted_dog']=row['p2']
          elif((row['p3_score']>row['p1_score']) & (row['p3_score']>row['p3_score'])):
            df_image_pred_clean.loc[index,'predicted_dog']=row['p3']
```

```
In [0]: # TEST 14
        df_image_pred_clean['predicted_dog'].value_counts()
```

```
Out[0]:                               388
        golden_retriever              166
        Labrador_retriever            110
        Pembroke                       94
        Chihuahua                      94
        pug                            61
        toy_poodle                     48
        chow                           47
        Samoyed                        45
        Pomeranian                     41
        malamute                       32
        French_bulldog                 32
        cocker_spaniel                 31
        Chesapeake_Bay_retriever       30
        miniature_pinscher             26
        Cardigan                       22
        Staffordshire_bullterrier      22
        German_shepherd                21
        Eskimo_dog                     20
        Siberian_husky                 20
        Shih-Tzu                       20
        Rottweiler                     19
        Lakeland_terrier               18
        beagle                         18
        Shetland_sheepdog              18
        kuvasz                         18
        Maltese_dog                    18
        Italian_greyhound              17
        basset                         17
        West_Highland_white_terrier    16
                                     ...
        giant_schnauzer                 4
        Saluki                          4
        Gordon_setter                   4
        Ibizan_hound                    4
        Weimaraner                      4
        Afghan_hound                    4
```

```
Rhodesian_ridgeback             4
Leonberg                        3
cairn                           3
komondor                        3
Brabancon_griffon               3
toy_terrier                     3
briard                          3
Irish_water_spaniel             3
curly-coated_retriever          3
Scottish_deerhound              3
Greater_Swiss_Mountain_dog      3
Sussex_spaniel                  2
Appenzeller                     2
groenendael                     2
Australian_terrier              2
black-and-tan_coonhound         2
wire-haired_fox_terrier         2
Japanese_spaniel                1
standard_schnauzer              1
silky_terrier                   1
Irish_wolfhound                 1
EntleBucher                     1
clumber                         1
Scotch_terrier                  1
Name: predicted_dog, Length: 113, dtype: int64
```

```
In [0]: # DEFINE: 16 Tidiness Issue - Pull in the data from Image Prediction and Json data to th

        # CLEAN 16

        # Lets put all the data together in the twitter archive dataframe
        # Merge Archived tweet data with the downloaded Json data
        # Note, here we will want only the tweets which are not re-tweets which is what we have
        # Therefore use and innerjoin where entries from both twitter archive and json data matc
        df_master_data = pd.merge(df_twitter_arch_clean, df_jdata_clean, on='tweet_id', how='inn

        # For the image prediction merge, we will do a left join to get the details for the main
        df_master_data = pd.merge(df_master_data, df_image_pred_clean, on='tweet_id', how='left'
```

```
In [0]: # TEST 16
        df_master_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2121 entries, 0 to 2120
Data columns (total 41 columns):
tweet_id                            2121 non-null int64
in_reply_to_status_id               77 non-null float64
in_reply_to_user_id                 77 non-null float64
```

```
timestamp                          2121 non-null object
source                             2121 non-null object
text                               2121 non-null object
retweeted_status_id                0 non-null float64
retweeted_status_user_id           0 non-null float64
retweeted_status_timestamp         0 non-null object
expanded_urls                      2063 non-null object
rating_numerator                   2121 non-null int64
rating_denominator                 2121 non-null int64
name                               2121 non-null object
doggo                              2121 non-null object
floofer                            2121 non-null object
pupper                             2121 non-null object
puppo                              2121 non-null object
combined_puppy_stage               2121 non-null object
text_uppercase                     2121 non-null object
numerator_clean                    2121 non-null float64
denominator_clean                  2121 non-null float64
timestamp_clean                    2121 non-null datetime64[ns, UTC]
retweeted_status_timestamp_clean   0 non-null datetime64[ns]
name_clean                         2121 non-null object
retweet_count                      2121 non-null int64
favorite_count                     2121 non-null int64
jpg_url                            1940 non-null object
img_num                            1940 non-null float64
p1                                 1940 non-null object
p1_conf                            1940 non-null float64
p1_dog                             1940 non-null object
p2                                 1940 non-null object
p2_conf                            1940 non-null float64
p2_dog                             1940 non-null object
p3                                 1940 non-null object
p3_conf                            1940 non-null float64
p3_dog                             1940 non-null object
p1_score                           1940 non-null float64
p2_score                           1940 non-null float64
p3_score                           1940 non-null float64
predicted_dog                      1940 non-null object
dtypes: datetime64[ns, UTC](1), datetime64[ns](1), float64(13), int64(5), object(21)
memory usage: 696.0+ KB
```

```
In [0]: # DEFINE: 10. Tidiness Issue Lets drop some of the additional columns to make it look ti

        # CLEAN
        cols_to_drop = ['in_reply_to_status_id', 'in_reply_to_user_id', 'doggo', 'floofer', 'pup
        df_mastr_clean = df_master_data.copy()
        df_mastr_clean.drop(cols_to_drop, axis=1, inplace=True)
```

```
In [0]: # TEST
        df_mastr_clean.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 2121 entries, 0 to 2120
Data columns (total 21 columns):
tweet_id                2121 non-null int64
timestamp               2121 non-null object
source                  2121 non-null object
text                    2121 non-null object
rating_numerator        2121 non-null int64
rating_denominator      2121 non-null int64
name                    2121 non-null object
combined_puppy_stage    2121 non-null object
text_uppercase          2121 non-null object
numerator_clean         2121 non-null float64
denominator_clean       2121 non-null float64
timestamp_clean         2121 non-null datetime64[ns, UTC]
name_clean              2121 non-null object
retweet_count           2121 non-null int64
favorite_count          2121 non-null int64
jpg_url                 1940 non-null object
img_num                 1940 non-null float64
p1_score                1940 non-null float64
p2_score                1940 non-null float64
p3_score                1940 non-null float64
predicted_dog           1940 non-null object
dtypes: datetime64[ns, UTC](1), float64(6), int64(5), object(9)
memory usage: 364.5+ KB
```

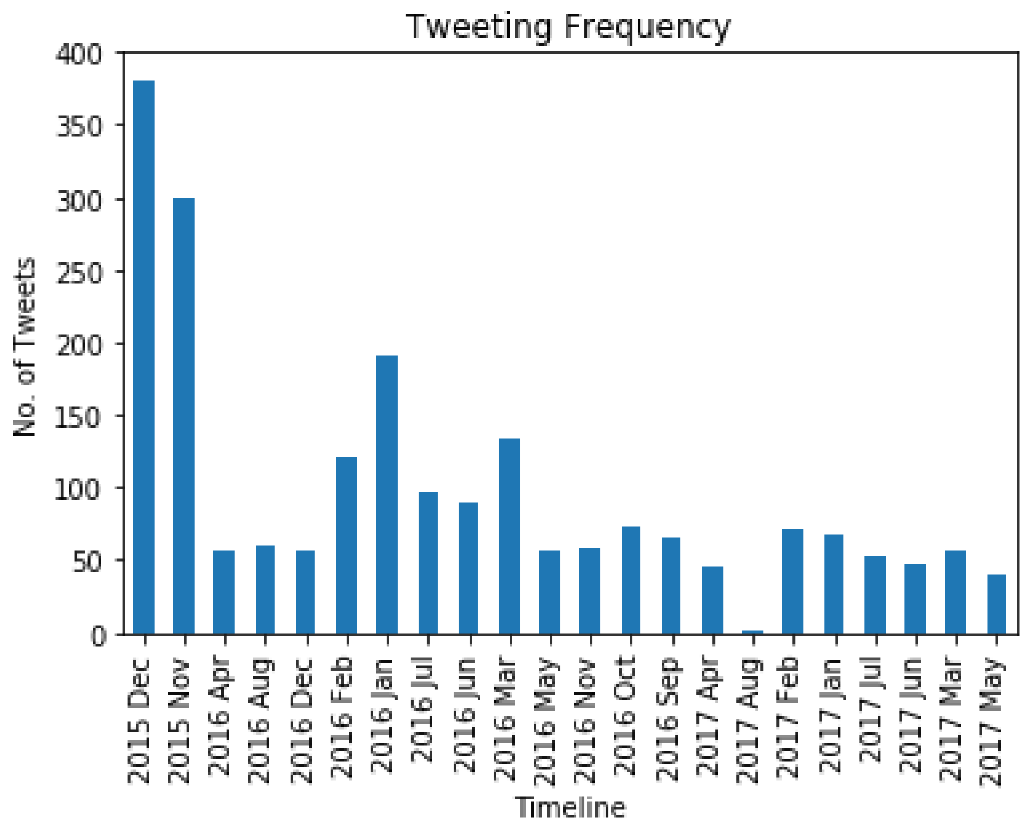
```
In [0]: df_mastr_clean.head()

Out[0]:              tweet_id                   timestamp  ...  p3_score      predicted_dog
        0  892420643555336193  2017-08-01 16:23:56 +0000  ...  0.000000
        1  892177421306343426  2017-08-01 00:17:27 +0000  ...  0.068957          Chihuahua
        2  891815181378084864  2017-07-31 00:18:03 +0000  ...  0.031379          Chihuahua
        3  891689557279858688  2017-07-30 15:58:51 +0000  ...  0.000000  Labrador_retriever
        4  891327558926688256  2017-07-29 16:00:24 +0000  ...  0.175219             basset

        [5 rows x 21 columns]
```

5 Analyze and visualize - three (3) insights and one (1) visualization

```
In [0]: # What is the average dog rating
        df_mastr_clean.numerator_clean.mean()

Out[0]: 12.161848184818481
```

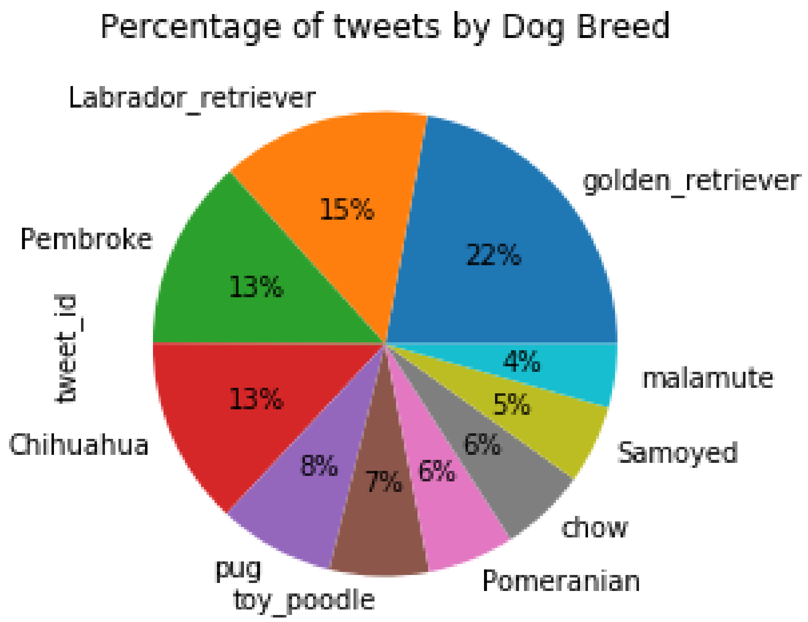
```
In [0]: # What is the average rating by dog type as based on the image predictions
        df_mastr_clean[df_mastr_clean.predicted_dog!=''].groupby('predicted_dog')['numerator_cle
        # Insight 1: This shows that the dog type Clumber most probably has the highest average

Out[0]: predicted_dog
        Saluki    12.5
        Name: numerator_clean, dtype: float64

In [0]: df_mastr_clean[df_mastr_clean.predicted_dog!=''].groupby('predicted_dog')['numerator_cle
        # Insight 1 contd. the dog type Japanese Spaniel most probably has the lowest average ra

Out[0]: predicted_dog
        Japanese_spaniel    5.0
        Name: numerator_clean, dtype: float64

In [0]: # Insight 2: Which type of dogs were retweeted the most

        df_mastr_clean[df_mastr_clean.predicted_dog!=''].groupby('predicted_dog')['retweet_count

        # Bedlington Terrier dogs were the most re-tweeted

Out[0]: predicted_dog
        Bedlington_terrier    8740.2
        Name: retweet_count, dtype: float64

In [0]: # Insight 2: Contd.: Which type of dogs were retweeted the least
        df_mastr_clean[df_mastr_clean.predicted_dog!=''].groupby('predicted_dog')['retweet_count

        # groenendael dogs were the least re-tweeted

Out[0]: predicted_dog
        groenendael    276.5
        Name: retweet_count, dtype: float64

In [0]: # Insight 3: Which type of dogs were favorited the most
        df_mastr_clean.groupby('predicted_dog')['favorite_count'].mean().sort_values(ascending=F

        # Saluki dogs were the most favorited

Out[0]: predicted_dog
        Bedlington_terrier    24438.4
        Name: favorite_count, dtype: float64

In [0]: # Insight 3: Contd: Which type of dogs were favorited the least
        df_mastr_clean.groupby('predicted_dog')['favorite_count'].mean().sort_values(ascending=T

        # Brabancon_griffon dogs were the least favorited

Out[0]: predicted_dog
        Brabancon_griffon    885.0
        Name: favorite_count, dtype: float64
```

```
In [0]: #Insight 4: Which is the most common dog name
        df_mastr_clean['name_clean'].value_counts().sort_values(ascending=False).head(5)

        # Charlie and Lucy seems to the most common dog names followed by Cooper and Oliver

Out[0]:             731
        Charlie      11
        Lucy         11
        Oliver       10
        Cooper       10
        Name: name_clean, dtype: int64

In [0]: #Insight 5: which is the most favorited dog stage
        df_mastr_clean.groupby('combined_puppy_stage')['favorite_count'].mean().sort_values(asce

        #puppo seems to be most popular

Out[0]: combined_puppy_stage
        doggo,puppo    47844.0
        Name: favorite_count, dtype: float64

In [0]: # When was the first tweet and the last tweet in this dataset
        df_mastr_clean.timestamp_clean.min()

Out[0]: Timestamp('2015-11-15 22:32:08+0000', tz='UTC')

In [0]: df_mastr_clean.timestamp_clean.max()

Out[0]: Timestamp('2017-08-01 16:23:56+0000', tz='UTC')

In [145]: #Visualization 1
          # What was the frequency of tweets based on this dataset
          ax = df_mastr_clean.groupby(df_mastr_clean['timestamp_clean'].dt.strftime('%Y %b'))['t
          ax.set_title('Tweeting Frequency')
          ax.set_xlabel('Timeline')
          ax.set_ylabel('No. of Tweets')

Out[145]: Text(0, 0.5, 'No. of Tweets')
```



```
In [146]: # Visualization 2
          # Which type of dog was the most tweeted about. Since there are many dog types
          # Lets pare it down to the top 10 most frequently tweeted dog type
          df_tweet_countby_dogtype = df_mastr_clean[df_mastr_clean['predicted_dog']!=''].groupby
          df_tweet_countby_dogtype

Out[146]: predicted_dog
          golden_retriever      150
          Labrador_retriever     99
          Pembroke               90
          Chihuahua              89
          pug                    55
          toy_poodle             47
          Pomeranian             41
          chow                   41
          Samoyed                37
          malamute               30
          Name: tweet_id, dtype: int64

In [144]: # lets do a pie chart for this information
          ax = df_tweet_countby_dogtype.plot.pie(autopct='%1.0f%%')
          ax.set_title('Percentage of tweets by Dog Breed')
```

```
Out[144]: Text(0.5, 1.0, 'Percentage of tweets by Dog Breed')
```



```
In [0]: df_mastr_clean.to_csv(fn_final_clean_file, index=False)

In [0]:
```