Introduction

Following project can be seen in 3 distinct parts.

- Data Gathering
- Data Assessment
- Data Cleaning

Data Gathering

It was done in three steps. Namely, downloading readily available data, programmatically downloading data hosted on Udacity server and finally, downloading data via twitter API. Three data files were gathered, downloaded and accumulated.

1. Twitter_archive_enhanced.csv (readily available) to be loaded as pandas dataframe with read_csv method
2. image_predictions.tsv containing tweet image predictions about dog breed and confidence levels to loaded as pandas dataframe using Requests library.
3. tweet_json.txt as a result of querying twitter API based on unique tweet ID from archive file.

Data Assessment

After gathering and storing the data to text files and reading back them into dataframes, we start assessment. Before diving into, let’s merge archive_df and tweet_df for sake of clarity and reduced efforts afterwards. Eventually, we will merge them anyway.

1. Check for data types
   a) date and timestamp related columns are not in the datetime objects
2. Check for retweets
   a) rows where *retweeted_status_id* is not null are the retweets
   b) there are 168 such rows
3. Check for unique dog names (and count)
   a) the dog names extraction appear to be not accurate as we see many dog names which are regular English language words

    b) upon further inspection whether dog name is indeed not present in a tweet or extraction failed, it turns out that there are tweets where the dog names are present but didn’t get extracted. e.g. tweet at row: 839
    c) dog names are in upper case, meaning all lowercase dog names are not “real” dog names
4. Check for records without images
    a) records without images (i.e. *expanded_urls* is null) can be treated as unusable records
5. Check for rating denominator
    a) rating denominator is usually 10 or multiples of 10s when there are more dogs in the picture
    b) any records where denominator is not multiple of 10 is suspicious and indeed checking for those reveals that extraction was faulty
6. Check for float numerator
    a) some records have float numerators. This could be either due to faulty extraction or

At the end of the assessment, it can be boiled down to following Quality and Tidiness issues:

- Quality Issues
    - Incorrect dtype to columns in *archive_tweet_df*. e.g. *timestamp* -> datetime
    - Records representing retweets
    - Subsequently, columns related to *retweeted_status_**
    - Records with no images (*expanded_url*)
    - Float numerator in the text not correctly represented in column *rating_numerator*
    - denominator that are not 10 (or multiples of 10)
    - *tweet_id: 810984652412424192* does not have rating
    - Incorrect dog names
    - No data values in dog names columns are 'None' as string dtype
    - No data values as None in doggo, floofer, pupper, puppo

- Tidiness Issues
  - consolidate doggo, floofer, pupper, puppo to one column *dog_stages*
  - potentially only keep *p1*, *p1_conf* and *p1_dog* columns in *prediction_df*
  - merge *archive_df* and *tweet_df* (already done!)

Data Cleaning

- We start with assigning correct datatypes to columns
  - *timestamp* → datetime
  - *rating_numerator* → float
- As a starter, the rows and columns corresponding to retweets were removed along with records with no images.
- To tackle float numerator incorrectly represented in the column as opposed to in text. The reason could be the text parsing algorithm only consider first occurrence of a number and considered as numerator. We can use regular expressions to extract and verify that. Logic:
  - extract list of *tweet_id*s to fix
  - extract correct numerators from text
  - update records
- Upon checking, we can confirm that these numerators were correctly extracted and dataframe was updated
- Denominators that are not 10 or multiples of 10 was investigated by checking the text. One example of such case is presented below where:

666287406224695296	This is an Albanian 3 1/2 legged Episcopalian. Loves well-polished hardwood flooring. Penis on the collar. 9/10 https://t.co/d9NcXFKwLv
740373189193256964	After so many requests, this is Bretagne. She was the last surviving 9/11 search dog, and our second ever 14/10. RIP https://t.co/XAVDNDaVgQ

- In both the tweets above we can see that there is an occurrence of *x/x* pattern which could have been the reason for incorrect extraction. Since we have only 3 cases, we can fix this manually rather quickly by simply reassigning the numbers to the dataframe based on corresponding *tweet_id*.

Test shows that the original query only returns one tweet where there was no rating present in the original text

- Above investigation gave us *tweet_id* (810984652412424192) where rating was not present in text and it only makes sense to remove it
- Next gigantic effort is to fix incorrect or missing dog names. We can search in the tweet text to retrieve this info. In our case, 5 different strategy that can be deployed where we extract text where name is:
    - lowercase and text contains 'name is'
    - None and text contains 'name is'
    - lowercase and text contains 'named'
    - None and text contains 'named'
    - lowercase and text contains 'That is'
    - 'O' instead of O'Malley
- The logic for all the cases remain the same as following:
    - extract text columns
    - get tweet_ids to fix
    - extract dog name from text
    - fix with ids
- Finally, we replace string None to *np.NaN* for no data values
- Replace None in doggo, floofer, pupper, puppo with empty string
- Consolidate 'doggo', 'floofer', 'pupper', 'puppo' to single column dog_stage, drop them and replace no data values to NaN
- Merge all the dataframes into one master dataframe