Introduction

Real-world data rarely comes clean. Using Python and its libraries,we will gather data from a variety of sources and in a variety of formats, assess its quality and tidiness, then clean it. This is called *data wrangling*.

The dataset that we will be wrangling (and analyzing and visualizing) is the tweet archive of Twitter user @dog_rates, also known as WeRateDogs. WeRateDogs is a Twitter account that rates people's dogs with a humorous comment about the dog. These ratings almost always have a denominator of 10. The numerators, though? Almost always greater than 10. 11/10, 12/10, 13/10, etc. Why? Because "they're good dogs Brent." WeRateDogs has over 4 million followers and has received international media coverage.

Step 1:Gather

In this step,we will obtain data from three sources

- The WeRateDogs Twitter archive which has been provided to us which contains basic tweet data for all 5000+ of their tweets
- The tweet image predictions that contains image predictions that classify dog breeds
- Additional data from the Twitter API that contains retweet count and favorite count

Step 2:Assess Data

When assessing, you're inspecting your data set for two things:

- Data quality issues:
    - **Completeness**: do we have all of the records that we should? Do we have missing records or not? Are there specific rows, columns, or cells missing?
    - **Validity**: we have the records, but they're not valid, i.e., they don't conform to a defined schema. A schema is a defined set of rules for data. These rules can be real-world constraints (e.g. negative height is impossible) and table-specific constraints (e.g. unique key constraints in tables).

    - **Accuracy**: inaccurate data is wrong data that is valid. It adheres to the defined schema, but it is still incorrect.
    - **Consistency**: inconsistent data is both valid and accurate, but there are multiple correct ways of referring to the same thing. Consistency, i.e., a standard format, in columns that represent the same data across tables and/or within tables is desired.
- Lack of tidiness: Data that has specific structural issues that slow you down when cleaning and analyzing, visualizing, or modeling your data later.
    - Each variable forms a column
    - Each variable forms a row
    - Each type of observational unit forms a table

Data Quality Issues

1. We have inconsistent naming format in the images dataset with the some dog names capitalised and the rest are in lowercase
2. We have duplicated expanded urls originating from retweets and one from an original tweet
3. We have missing data in the archive dataset with only 'Expanded URLS' and 'names' column worth deep diving into as the rest of the columns with missing data are insignificant.However we will not drop the missing names columns as the text column has some of the dog names and can be extracted for this.
4. We have inconsistent naming format in the images dataset with the some dog names capitalised and the rest are in lowercase
5. We have invalid dog names such as 'a' ,'the','o' and 'an' in the archive dataset
6. Inconsistent data types of datetime objects and tweet_ids should be converted to string objects
7. Prediction columns have inconsistent naming formats
8. Duplicated jpg_urls which should be removed
9. Convert NaN to None

Tidiness/Structural Issues

1. The three datasets should be combined into one

2. The columns doggo, floofer, pupper, puppo can be melted to form a single column to indicate the dog_stage
3. The text column in the archive contains pronouns which can be extracted to get the dog's gender
4. We can extract the dog breed from the prediction columns and melt into one
5. Remove unnecessary columns

Step 3:Cleaning Data

In this step we are are going to

- Define
- Code
- Test

But first we make a copy of our original datasets and clean the data quality issues as well as structural issues

1.Merge the datasets into one

```
master_df=pd.merge(pd.merge(archive_clean,tweets_clean,on='tweet_id'),images_clean,on='tweet_id')
```

2.Removing duplicated data from the expanded_urls column

Merging the datasets takes care of this

3.Change the dognames to a consistent format i.e to capitalize all names

```
master_df_clean['name']=master_df_clean.name.str.capitalize()
```

4.Remove invalid dog names and replace with None

```
master_df_clean=master_df_clean.replace({'name':{'A':'None','An':'None','The':'None','O':'None'}})
```

5.Extract dog gender from text column

```
# Looping for pronouns of male or female;

male_pronouns = ['He', 'he','HE','HIM', 'him','Him','HIS', 'his','His',"HE'S",'he's',"He's",'HIMSELF', 'himself','Himself','BOY','boy','Boy']
female_pronouns = ['She', 'she', 'SHE','HER','HERS','her','Her','hers','Hers','HERSELF', 'herself','Herself','she's',"She's","SHE'S",'SHE','GIRL','Girl','girl']

gender = []

for t in master_df_clean['text']:
  if any(map(lambda v:v in male_pronouns, t.split())):
    gender.append('male')
  # Female
  elif any(map(lambda v:v in female_pronouns, t.split())):
    gender.append('female')
  # If neither male nor female then none
  else:
    gender.append('None')

#add a new column for dog gender
master_df_clean['gender'] = gender
```

6.Correct incorrect dtypes

In [47]:

```
from datetime import datetime
```

```
#Convert tweet_id,id,id_str to string and timestamp to datetime object
master_df_clean['tweet_id']=master_df_clean['tweet_id'].astype(str,errors='ignore')
master_df_clean['id']=master_df_clean['id'].astype(str,errors='ignore')
master_df_clean['id_str']=master_df_clean['id_str'].astype(str,errors='ignore')
master_df_clean['timestamp']=pd.to_datetime(master_df_clean['timestamp'],errors='ignore')
```

7.Extract doggo,floofer,pupper,puppo from text column into one

```
stages= ['doggo', 'floofer', 'pupper', 'puppo']
for stage in stages:
  master_df_clean[stage] = archive_clean[stage].apply(lambda x: np.NaN if x == 'None' else x)

master_df_clean['life_stage'] = master_df_clean[['doggo', 'floofer', 'pupper', 'puppo']].astype(str).sum(1)
master_df_clean['life_stage'] = master_df_clean['life_stage'].apply(lambda x: x.replace('nan', ''))
master_df_clean['life_stage'] = master_df_clean['life_stage'].apply(lambda x: np.NaN if x == '' else x)

master_df_clean['life_stage'].replace({'doggopupper':'doggo pupper',
                'doggofloofer':'doggo floofer',
                'doggopuppo':'doggo puppo',
                '':None}, inplace=True)
```

8.Clean the prediction columns names and combine into one

```
master_df_clean.rename(columns={'p1':'first_prediction',
'p1_conf':'first_prediction_confidence', 'p1_dog':'is_first_prediction_dog',
                'p2':'second_prediction','p2_conf':'second_prediction_confidence',
'p2_dog':'is_second_prediction_dog',
                'p3':'third_prediction','p3_conf':'third_prediction_confidence',
'p3_dog':'is_third_prediction_dog'},inplace=True)

#Pick the first true dog_breed precidiction together with the confidence interval
prediction_breed = []
confidence = []

def get_prediction_confidence(dataframe):
  if dataframe['is_first_prediction_dog'] == True:
    prediction_breed.append(dataframe['first_prediction'])
    confidence.append(dataframe['first_prediction_confidence'])
  elif dataframe['is_second_prediction_dog'] == True:
    prediction_breed.append(dataframe['second_prediction'])
    confidence.append(dataframe['second_prediction_confidence'])
  elif dataframe['is_third_prediction_dog'] == True:
    prediction_breed.append(dataframe['third_prediction'])
    confidence.append(dataframe['third_prediction_confidence'])
  else:
    prediction_breed.append('None')
    confidence.append(0)

master_df_clean.apply(get_prediction_confidence, axis=1)
master_df_clean['prediction_breed'] = prediction_breed
master_df_clean['confidence'] = confidence
```

9.Capitalize dog breed names and remove the underscores

```
#Capitalize dog_breed
```

```
master_df_clean['prediction_breed']=master_df_clean['prediction_breed'].str.capitalize()

#Remove the underscores in the names
master_df_clean['prediction_breed']=master_df_clean['prediction_breed'].str.replace('_',' ')
master_df_clean['prediction_breed']=master_df_clean['prediction_breed'].str.replace('-',' ')
```

10.Remove duplicated jpg_urls

```
#Drop duplicates and keep the first occurences of the urls
master_df_clean=master_df_clean.drop_duplicates(subset=['jpg_url'],keep='first')
```

11.Convert NaN to None

```
#Replace NaN to None
master_df_clean=master_df_clean.replace(np.nan,'None')
```

12.Remove unnecessary columns

```
master_df_clean.drop(columns=['in_reply_to_status_id_x',
'in_reply_to_user_id_x','id', 'id_str', 'full_text', 'truncated',
    'display_text_range', 'entities', 'extended_entities', 'source_y','source_x',
    'in_reply_to_status_id_y', 'in_reply_to_status_id_str',
    'in_reply_to_user_id_y', 'in_reply_to_user_id_str',
    'in_reply_to_screen_name', 'first_prediction',
'first_prediction_confidence',
    'is_first_prediction_dog', 'second_prediction',
    'second_prediction_confidence', 'is_second_prediction_dog',
    'third_prediction', 'third_prediction_confidence',
    'is_third_prediction_dog','retweeted_status_id',
```

```
                    'retweeted_status_user_id','retweeted_status_timestamp',
'expanded_urls','user', 'geo',
    'coordinates','contributors', 'is_quote_status', 'possibly_sensitive',
'possibly_sensitive_appealable', 'lang', 'possibly_sensitive_appealable', 'lang',
    'retweeted_status', 'quoted_status_id', 'quoted_status_id_str',
    'quoted_status','place'],inplace=True)
```

Step 4: Storing Data

```
#Storing the clean dataset as a csv
master_df_clean.to_csv('twitter_archive_master.csv',index=False)
```