Wrangle Report

Gather

To gather the image predictions from the url provided I first used the requisitions library to save an access point to the url in a variable (response) and then used BeautifulSoup to parse the url, making sure to pass 'lxml' as an argument to avoid an error being thrown. After saving the Beautiful Soup result in a variable (master_data) I then use the following code to read the image predictions into a dataframe:

```
with open(url.split('/')[-1].replace('-', '_'), 'wb') as file:
    file.write(responce.content)

image_predictions = pd.read_csv(url.split('/')[-1].replace('-', '_'), sep='\t')
```

And check that this has functioned as expected by printing the head of the new dataframe image_predictions. Next I read the twitter-archive-enhanced.csv into a variable df and check this has functioned as expected using the head function. Following this I have saved all the necessary information concerning access to my Twitter API and then opened an outfile as a json file and using a for loop accessed the Twitter API and fetched all the data corresponding to retweet and favourite statistics. Following this I have read the json file into a dataframe (API_df) and checked using a fails dictionary, created during the same function where I was reading the API, to check how many entries were not able to be fetched (877).

Access

At the start of the access process print the three dataframes I have so far created to gather a broader picture as to what I am handling. I also use describe() and info() as well as duplicated() to understand the spread of data as well as dtypes and number of duplicates if any (luckily in this case none).

Through this process the following is what I observed:

Quality

All tweet_id's are integers rather than strings

Image_predictions

- Tweet ID is an int

`df`
- img_num is a irrelavant column in image_predictions
- p1-3_dog are irrelavent columns in image_predictions
- timestamp is a string
- retweeted_status_id is a float
- retweeted_status_timestamp is an object

- All retweets and replies should be deleted according to the prject ruburic.

Tidiness

Image_predictions

- floofer, pupper, puppo columns marked as either none or the name of the column
- ['p2', 'p2_conf', 'p3', 'p3_conf'] are all irrelavant in terms that we are only going to need to look at the predictors best guess
- ['p1_dog', 'p2_dog', 'p3_dog'] are all irrelavant columns with only one value present (thgat being 'True', this column will be not be useful in our analysis do best to delete.

Df

- denominator and numerator in seperate columns
- value inputs under ['doggo', 'floofer', 'pupper', 'puppo'] could all be in one column as in vast majority of cases when a value is entered in at least one of these columns it is not mentioned in any other ones. We can therefore combine these columns.

Clean

First thing I have done is to make a copy of each of the imnage_predictions and API_df dataframes. Next I have rectified both the instances of incorrect dtypes (timestamp and tweet_id) in the df and API_df dataframes respectively. I have then checked to make sure this has worked using the .info() function. Next I madesure the tweet id's from df matched up with the tweet id's in image_predictions. I have then done the same with the tweet id's obtained by the API and happily all match up. After these respective processes I have merged the dataframes together and used the .head() function to check all the columns have been added to df. I have then dropped ['p2', 'p2_conf', 'p3', 'p3_conf'] as I do not think the second or third choices of the neural network arerelavant to our dataset (I come back to this in the Act Report). I do the same with ['p1_dog', 'p2_dog', 'p3_dog'] and again come back to it in the Act report. Next I replace NanN values with the ocrect np.nan function so that tehse instances are easier to deal with within pandas. I also extract the the dog_type values and parse them into one column and also creating arating column with the numerator and denominator columns. Next I drop all comuns which are retweets or replies and remove the necessary columns, checking that this has worked with the .head() function. Luckily in my data there are no instances when the extracted dog_types double up on eachotehr as shown by my use of value_counts() within in[55].