WRANGLING EFFORTS

Upon manually downloading the twitter-archive-enhanced.csv, I opened it in Jupyter notebook for the future tasks to be performed. Then I downloaded the image_predictions.tsv file, which we were supposed to download programmatically, via the requests library. Then we were supposed to get the final dataset, the tweet information(tweetID, fav count and retweet count) for each tweetID that we had in our twitter-archive-enhanced.csv file using the twitter API(tweepy) and store that JSON data in another file tweet_json.txt.

After downloading the twitter_archive_enhanced.csv file, I opened it up for extracting the tweetID's so that I could download additional tweet information using tweepy. However, upon trying it, I realised that the tweetID is stored in float64 format(8.924210e+17) which could not be used as such. It was stored in this format in the csv file itself, so converting into int() also would not work(leading to loss in precision). There was only 1 way possible, extracting the tweetID from the http in expanded_urls column.

After extracting correct tweet_id from expanded_urls, I could download the tweet data. I downloaded each tweet's data using tweepy, and the tweet_id's. The response of get_status() in tweepy is a Status object, and the JSON data can be obtained by using a private attribute _json. This data was then written to a tweet_json.txt file, with each tweet data being stored in a new line.

Then I opened this file in read mode, read each line data using json.loads into a json object(python dictionary) and appended the dictionary with required attributes(retweet_count and fav_count) into a list. Then I converted this list of dictionaries into a dataframe df2.

After that, the gathering phase was over, and I started assessing the dataframes. I started assessing the expanded_urls column, and certain discrepancies started becoming visible. Some cells had multiple URL's, sometimes the same twitter address being replicated multiple times, sometimes 2 different addresses were stored in same cell, and sometimes an altogether different address (not twitter weRateDogs was stored). This issue had to be taken care of before the tweetID was extracted.

One way of doing this was to use regex to extract the correct http address from each cell in expanded_urls column. Upon a lot of visual and programmatic assessment, I realised that the only format acceptable was

https://twitter.com/dog_rates/status/<18 digit tweetID>/video/1 and
https://twitter.com/dog_rates/status/<18 digit tweetID>/photo/1

Any other address was either not of twitter, or did not belong to the WeRateDogs page. So ultimately using above regex and pandas extract function, I could extract both the correct http addresses and the tweetID's.

Now both expanded_urls and tweet_id columns have quite a few NaN's in them. Now in order to get rid of these rows, it is needed to obtain the index of these cells. So, using pandas function isna() I obtained

the index of the cells, then I extracted the indices of the rows have nan in tweet ID and http address and then dropped those rows.

Upon further observation, I realised that many tweet ID's are repeated. These duplicate tweetID's need to be removed before the additional tweet information is gathered and integrated into the dataset, as it will lead to redundancy. So using drop_duplicates() of pandas, I removed all second occurrences of the same tweet ID's.

The timestamp and retweeted timestamp are stored in object format, so it makes sense to convert it into pandas datetime object.

Also, upon doing programmatic assessment, I realised that in dog names quite a few of them have name as 'a'. Now of course a dog name cannot be 'a', so most probably it is an input error. Since it is wrong data, it is better to convert it into 'None'.

Apart from that, there are many rows in pred dataframe which don't correspond to that of a dog, according to the ANN classifier. So these rows are to be removed. I kept only those rows for which the top prediction is dog, or if both the 2nd and 3rd highest prediction is that of a dog, since upon going to the given links, I realised that the top prediction does get wrong sometimes.

I created a new column which will store the dog breed of the highest confident prediction. If the top prediction was that of a dog, then the breed would take the classification of the top prediction, else it will take the classification of the second prediction.

Removed all other information from pred dataframe, kept only the dog classification, tweet_id and jpg_url. Then I merged this dataframe into the main dataframe df, using tweet_id as the common column. After that, I merged the df2 dataframe on df using the same column tweet_id, and stored this master dataframe in a file master_dataframe.csv.