Data accessing

The first dataset twitter_archieve_enchanced data is download and access used pandas library
The second dataset is download using requests package. the link and downloaded programmatically and stored in particular folder
The third dataset is download from Twitter, creating a twitter developer account and download file tweet_json converting into csv using JSON module data/tweet_json.txt using the JSON library and the open command as can be seen at In [5] in the ipynb/HTML file. The problem encountered here was that I wasn't using the JSON library at first which made writing the file a little more difficult than it needed to be. I had to make sure to add wait_on_rate_limit=True, wait_on_rate_limit_notify=True into the API constructor within the code to make sure that the connection would not time out from the server.

Programmatic assessment

The programmatic assessment is where I found most of the issues that I wanted to address during the wrangling phase. It is worth noting however that there is not a hard line between what constitutes programmatic and visual, as I noted within the notebook itself. The general problems that I found were missing values using pandas.DataFrame.info(), which I used quite a lot throughout the wrangling process. The tweet_id was always an integer when it should have been a string (see Ordinal vs Categorical values).

I used pandas' filtering capabilities to help me figure out where to draw the line between which pictures were and weren't dogs. This means that I had to look at the top three predictions that were either predicted to be a dog or

not and then look at the image visually in order to assert if this categorization was true or not. I found that images were unlikely to dog if the top three predictions were false or if there was a True prediction with a probability of 0.2.

Data visualization and data cleaning

Next step is clean and visualizes the data, the imported data needed to be assessed for quality issues. This was done programmatically within my ipynb file. Visual assessment was not very well documented not only because it is difficult to point directly to what you are viewing without messing around with image files, but also because of I kind of neglected this part of the assessment a little bit. This meant that I ran into more problems during the cleaning stage later in the wrangling process.

I solved the problems that I outlined in the Visual Assessment. Some of the issues I had were due to not thoroughly assessing the data visually. This meant that I had to skip a lot throughout the document to make sure all data entries were properly updated, which was cumbersome.

There were also a lot of times when commands that I was using weren't working for one reason or another. One time I tried to remove duplicate rows using pandas.DataFrame.drop_duplicates() but this did not work because the rows that sometimes contained the same names of dogs sometimes had different tweet_ids.

However, having defined the small "step-by-step" tasks helped me a lot to focus on each individual one and execute them, which meant that this step took much less time than it might have if I did not have a plan, so I am very happy about that.

Data Saving

After the data cleaning was completed, I created a CSV file and an SQLite database using pandas and sqlalchemy respectively.

The SQLite database was made by creating an engine object using the sqlalchemy.create_engine() constructor and then using the pandas.DataFrame.to_sql() command using the engine I had just created to export the data.

The CSV creation was much less interesting, that was just done using pandas.DataFrame.to_csv().