Wrangle Report

Gathering

This was by far the most time-consuming part of the wrangling process for me. While accessing a file on hand and downloading a file programmatically were easy enough, I had big problems with the Twitter API.

First of all, I could download all columns in JSON format but the API call limit would be reached and I would retrieve only ~850 rows each time. In the project Slack channel, I found some people who experienced the same problem and they advised using the "wait_on_rate_limit=True" parameter in the API call method. Doing so I could download ~2,100 rows, however the data values were missing when I added the parameter. I tried not saving the JSON values directly into a text file, but storing them in an array which I would then save as a file. My lack of coding experience brought up some roadblocks in doing this, and I was unable to generate a dataframe with just the "tweet_id", "favorite_count" and "retweet_count". I went back to the original method of writing JSON values directly, and for some reason found that I could save the correct values for ~2,100 rows and create the necessary dataframe. Maybe I was missing a comma somewhere, I still don't know why it started working.

The problem is that due to the Twitter API limit, attempting to get the JSON values took about 45 minutes each time. Due to a busy schedule, this meant any unsuccessful attempts would result in postponing to another day. It was very frustrating.

Assessing

Assessing wasn't so bad. I used a preliminary programmatic assessment to make sure there were no duplicate tweet IDs and the data types were correct. The data types for "timestamp", "favorite_count" and "retweet_count" had to be changed.

The visual assessment was also fine. There were three predictions for dog breed and some were not even dogs, so I decided to just use the most likely dog prediction if there was one within the three available predictions.

Cleaning

I'm not the strongest coder, so I tried to keep the code as simple as possible. The toughest cleaning code here was in tidying the four "dog age" columns to one. I could have tried to use a for loop and pick the "dog age" value from the 4 columns if it was not "None" and then populate a newly created column with that, but I wanted to try a different way.

Since I was stuck on how to do this, I posed a question on Stack Overflow for the first time, which was a good experience.
https://stackoverflow.com/questions/52328474/python-set-new-column-value-if-multiple-columns-in-a-dataframe-have-any-value-o

I eventually settled on the following:

```
values = ['doggo', 'floofer', 'pupper', 'puppo']

tweets_clean['dog_age'] = tweets_clean[values].replace('None',np.nan).ffill(axis=1).iloc[:,-1].fillna('Unknown')
```

1) Replace “None” with missing values
2) Forward fill missing values
3) Select last column with iloc
4) Fill missing values with Unknown