Wrangling efforts

This project was probably the most time consuming one so far. Gathering the data worked pretty well as I am used to querying APIs from my day job. However, in the beginning I encountered some issues with erroneous encoding when writing the tweets to txt. After re-arranging the code and loading the tweets line by line into the dataframe that worked out fine. To be sure that I don't get any wrong results in the following, I decided to move forward with the provided tweet-json.txt (my own file is called with an underscore: tweet_json.txt).
After visually and programmatically assessing the data, I discovered several issues with regard to quality and tidiness. First of all I needed to make sure that only original tweets and no replies or retweets were part of the dataset. Moreover several minor issues with regard to data quality were obvious, like wrong entries for the name columns (starting with small letters instead of capitalized ones). Because some ratings also seemed off, I decided to re-pull the ratings from the original tweets and replaced those with the ones provided in the file. I used regex and lambda expressions to find the relevant parts in the strings. Moreover I fixed some incorrect data types, varying writing styles (capital vs. small letters) and None-types being filled in with string values.
Cleaning the tidiness issues was the harder challenge this time around. The dog types like ‚pupper', ‚floofer', etc. were spread across several columns. In a tidy data set, this should be contained in only one column stating which of the types a certain dog is assigned to. First I thought to go ahead with pd.melt. However that did not work out as I was not having any value column to work with. So I did some research on the web and found an actually pretty simple solution by joining the columns and replacing na values with blanks. I could have manually cleaned the rows where two dogs were assigned. However the task stated not to clean everything, so I decided to leave these values as they were as I was not planning to do any further analysis on this column in the following.
Then I tried to figure out how to proceed with the prediction file. I decided to go for a waterfall kind of approach: checking whether the first prediction was dog=True, if yes, take it, if not, proceed to prediction 2 and check if dog=True, if False proceed to prediction 3. If none was predicted a dog, I kept False as a value. So I ended up with having the predicted dog breed and confidence for the prediction per each tweet, pretty neat.
I decided to keep all data in one dataframe as splitting them as my main interest was tweet based. Therefore I think that the information belongs together as one piece, enriching the tweets initial data with further information.
In the analysis part I was interested in three main questions: Which dog breed was predicted with the highest confidence by the neural network? What is the most common length of a dog name within the data set? And: Which (predicted) dog breed received the highest rating on average and how far do the ratings spread per breed? The first two questions were analyzed by pure python. For the third one I decided to plot a Seaborn graph. I found an inspiring example on the Seaborn website that I wanted to adapt for my purposes. As the dataset is pretty cluttered, I decided to only use dog breeds for which I had more than 10 entries in the dataset in order for the results to be more meaningful. Also numerators higher than 20 were excluded from the visualization as the spread would be too big and distort the visualization. The dog breeds are ordered by their mean numerator ratings, moreover the spread of numerator ratings is included per dog breed.