Project2: Wrangling and Analyzing Data

Table of Contents

Introduction

This second project has to do with data wrangling. Wrangling Data includes; Data gathering, Assessing data and cleaning data. In this project the data were gathered using three methods; loading using pandas, using the request library and programmatically from tweet's JSON data.

Step1: Gathering Data

The first thing is to import all the necessary python libraries to be used for the project. The code for importing relevant libraries is shown below.

```
# Import the libraries
import pandas as pd
import requests
import tweepy
from tweepy import OAuthHandler
import json
from timeit import default_timer as timer
import os
import numpy as np
import re
%matplotlib inline
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

Next is to load all the dataset that will be used from various sources.

1. Loading WeRateDogs Twitter archive data (twitter_archive_enhanced.csv).

```
# Loading the first dataset directly
df_1= pd.read_csv('twitter-archive-enhanced.csv')
```

2. Using the Request library to download the tweet image prediction (image_predictions.tsv). It was hosted on Udacity's servers in tsv format and had to be downloaded using the requests library.

```
# Loading the second dataset using the request library
# df_2 for second DataFrame

df_2 = 'WeRateDogs_reviews'
if not os.path.exists(df_2):  # to check a folder exists or not
    os.makedirs(df_2)         # to create a directory/folder

url = 'https://d17h27t6h515a5.cloudfront.net/topher/2017/August/599fd2ad_image-predictions/image-predictions.tsv'

response = requests.get(url)
```

```
# Response [200] shows there is no error
response
```

```
<Response [200]>
```

```
with open(os.path.join(df_2, url.split('/')[-1]), mode = 'wb') as file:  #  extracting (image-predictions.tsv)
    file.write(response.content)
```

```
os.listdir(df_2)
```

```
['image-predictions.tsv']
```

```
# assigning df2 to the second dataset
df2 = pd.read_csv('image-predictions.tsv', sep='\t')
```

3. Loading the json file (tweet-json.txt) into a DataFrame.

```
# converting the text file to a data list and DataFrame
df3_list = []  # creat an empty list for the dataframe

with open ('tweet-json.txt', encoding = 'utf-8') as file:
    for line in file:
        parse_json_to_dic = json.loads(line)  # parse jason string to python dictionary
        df3_list.append(parse_json_to_dic)
```

```
# passing the tweet_id,favorites_count and retweet_count into Dataframe

# creating the three list that will serve as column header
idlist = []
retweetlist = []
favoritelist = []

for info in df3_list:
    i_d = info['id']
    retweets = info['retweet_count']
    favorites = info ['favorite_count']


    idlist.append(i_d)
    retweetlist.append(retweets)
    favoritelist.append(favorites)
# create twwet_df dataframe
    df_3 = pd.DataFrame({'tweet_id': idlist, 'retweet_count': retweetlist, 'favorite_count':favoritelist})
```

Step 2: Assessing Data

Three dataframe were created, df_1, df_2 and df_3.

Assessment Summary:

Quality

(First DataFrame = df_1)

(1) Validity:The datatype in the timestamp column should be Datetime.

(2) Accuracy:Text cloumn contains tiny url and urls.

(3) Accuracy: Names like 'None', 'a' and 'an' are not real names.

(4) Accuracy: The statistics shows that rating denominator has low value(s) equal to 0 and as high as 170.

(5) Consistency: Retweets (reply to tweets-RT) are not needed.

-In the project's requirements, only original ratings that have images are required, not retweets nor replies.

(6) Consistency:The columns 'in_reply_to_status_id', 'in_reply_to_user_id', 'retweeted_status_id', 'retweeted_status_user_id', and 'retweeted_status_timestamp' are irrelevant and somehow repititive, and should be removed.

(7) Accuracy: The numerator has values as low as 0 and as high as 1776.

(Third DataFrame = df_3)

(8) Validity: tweet_id is integer instead of string datatype

Tidiness

Variable should form a column

(1) The “doggo”, “floorfer”, “pupper”, and “puppo” columns are phases in a dog's development, they can combine into one column (dog_phase).

An observational unit should form a table

(2) All the three tables/dataframe should be combined not a dataframe since they represent the same observation.

The requirements of this project are only to assess and clean at least 8 quality issues and at least 2 tidiness issues in this dataset.

Quality Issues

Visual Assessment of df_1

(1) Some names are just a single letter e.g 'a', and word like 'none' in the name column.
(2) Text column contains urls.

```
df_1.head(4)
```

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text	retweeted_status_id	retweeted_status_user_id	retweeted_status_timestamp	
0	892420643555336193	NaN	NaN	2017-08-01 16:23:56 +0000	<a href="http://twitter.com/download/iphone" r...	This is Phineas. He's a mystical boy. Only eve...	NaN	NaN	NaN	https://twitter.com/do
1	892177421306343426	NaN	NaN	2017-08-01 00:17:27 +0000	<a href="http://twitter.com/download/iphone" r...	This is Tilly. She's just checking pup on you....	NaN	NaN	NaN	https://twitter.com/do
2	891815181378084864	NaN	NaN	2017-07-31 00:18:03 +0000	<a href="http://twitter.com/download/iphone" r...	This is Archie. He is a rare Norwegian Pouncin...	NaN	NaN	NaN	https://twitter.com/do
3	891689557279858688	NaN	NaN	2017-07-30 15:58:51 +0000	<a href="http://twitter.com/download/iphone" r...	This is Darla. She commenced a snooze mid meal...	NaN	NaN	NaN	https://twitter.com/do

```
df_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2356 entries, 0 to 2355
Data columns (total 17 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   tweet_id                    2356 non-null   int64  
 1   in_reply_to_status_id       78 non-null     float64
 2   in_reply_to_user_id         78 non-null     float64
 3   timestamp                   2356 non-null   object 
 4   source                      2356 non-null   object 
 5   text                        2356 non-null   object 
 6   retweeted_status_id         181 non-null    float64
 7   retweeted_status_user_id    181 non-null    float64
 8   retweeted_status_timestamp  181 non-null    object 
 9   expanded_urls               2297 non-null   object 
 10  rating_numerator            2356 non-null   int64  
 11  rating_denominator          2356 non-null   int64  
 12  name                        2356 non-null   object 
 13  doggo                       2356 non-null   object 
 14  floofer                     2356 non-null   object 
 15  pupper                      2356 non-null   object 
 16  puppo                       2356 non-null   object 
dtypes: float64(4), int64(3), object(10)
memory usage: 313.0+ KB
```

- Names like 'None', 'a' and 'an' are not real dog names.

```
# To check the series of names present in name column
df_1['name'].value_counts()[:20]
```

```
None       745
a           55
Charlie     12
Cooper      11
Lucy        11
Oliver      11
Tucker      10
Penny       10
Lola        10
Winston      9
Bo           9
Sadie        8
the          8
Daisy        7
Buddy        7
Toby         7
an           7
Bailey       7
Leo          6
Oscar        6
Name: name, dtype: int64
```

```
# checking the statistics of numerator and denomenator since they are the only real integers
df_1[['rating_numerator','rating_denominator']].describe()
```

	rating_numerator	rating_denominator
count	2356.000000	2356.000000
mean	13.126486	10.455433
std	45.876648	6.745237
min	0.000000	0.000000
25%	10.000000	10.000000
50%	11.000000	10.000000
75%	12.000000	10.000000
max	1776.000000	170.000000

(5) The statistics above shows that rating denominator has value(s) equal to 0 and as high as 170.

(6) Also the numerator has values as low as 0 and as high as 1776.

```
# show the rows with a numerator of zero value
df_1.query('rating_numerator == 0')
```

	tweet_id	in_reply_to_status_id	in_reply_to_user_id	timestamp	source	text
315	835152434251116546	NaN	NaN	2017-02-24 15:40:31 +0000	<a href="http://twitter.com/download/iphone" r...	When you're so blinded by your systematic plag...
1016	746906459439529985	7.468859e+17	4.196984e+09	2016-06-26 03:22:31 +0000	<a href="http://twitter.com/download/iphone" r...	PUPDATE: can't see any. Even if I could, I cou...

Display where the rating_numerator is greater than 600.

```
# Get the row where the numerators are > or = 600 (which is extremely high)
df_1.query('rating_numerator >=600')
```

retweeted_status_user_id	retweeted_status_timestamp	expanded_urls	rating_numerator	rating_denominator	name	doggo	floofer
NaN	NaN	NaN	666	10	None	None	None
NaN	NaN	NaN	960	0	None	None	None
NaN	NaN	https://twitter.com/dog_rates/status/749981277...	1776	10	Atticus	None	None

The second dataset (image-predictions.tsv)

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075 entries, 0 to 2074
Data columns (total 12 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   tweet_id  2075 non-null   int64  
 1   jpg_url   2075 non-null   object 
 2   img_num   2075 non-null   int64  
 3   p1        2075 non-null   object 
 4   p1_conf   2075 non-null   float64
 5   p1_dog    2075 non-null   bool   
 6   p2        2075 non-null   object 
 7   p2_conf   2075 non-null   float64
 8   p2_dog    2075 non-null   bool   
 9   p3        2075 non-null   object 
 10  p3_conf   2075 non-null   float64
 11  p3_dog    2075 non-null   bool   
dtypes: bool(3), float64(3), int64(2), object(4)
memory usage: 152.1+ KB
```

The third dataset is df_3 dataframe (tweet-json.txt)

```
# Check a random entries of the third dataset
df_3.sample(20)
```

	tweet_id	retweet_count	favorite_count
2269	667495797102141441	294	565
1	892177421306343426	6514	33819
1965	673342308415348736	649	1362
894	759099523532779520	4813	16101
1857	675497103322386432	1443	3397
1256	710283270106132480	580	2308
1356	703382836347330562	1285	3837
660	791026214425268224	4858	0
580	800443802682937345	5068	0
2155	669583744538451968	1017	1587
2141	669970042633789440	65	317

Tidiness

The “doggo”, “floorfer”, “pupper”, and “puppo” columns are phases in a dog's development, they can combine into one column (named dog_phase)

```
# Check the fisrt 4 rows of the first dataset
df_1.head(4)
```

_user_id	retweeted_status_timestamp	expanded_urls	rating_numerator	rating_denominator	name	doggo	floofer	pupper	puppo
NaN	NaN	https://twitter.com/dog_rates/status/892420643...	13	10	Phineas	None	None	None	None
NaN	NaN	https://twitter.com/dog_rates/status/892177421...	13	10	Tilly	None	None	None	None
NaN	NaN	https://twitter.com/dog_rates/status/891815181...	12	10	Archie	None	None	None	None
NaN	NaN	https://twitter.com/dog_rates/status/891689557...	13	10	Darla	None	None	None	None

```
# display the last 5 rows in the third dataset
df_3.tail(5)
```

	tweet_id	retweet_count	favorite_count
2349	666049248165822465	41	111
2350	666044226329800704	147	311
2351	666033412701032449	47	128
2352	666029285002620928	48	132
2353	666020888022790149	532	2535

- All the three tables should be combined into a dataframe since they represent the same observation.

Clean (1): (Tidiness(2))

Define

- Merge all the three tables into one.

```
# Merge the 3 tables, first merge df_2 and df2, then with df3.
First_merge = df_1_clean.merge(df2_clean, on = 'tweet_id')  # merge df_1_clean and df2_clean, name it First_merge
master_df = First_merge.merge(df_3_clean, on = 'tweet_id')  # merge First_merge and df_3, name it master_df
```

Clean(2) : Quality(1)

Define

- Change Timestamp column to datetime datatype

Code

```
# Convert the timestamp column to datetime datatype
master_df.timestamp = pd.to_datetime(master_df.timestamp, format = '%Y-%m-%d %H:%M:%S')
```

Test

```
# check the datatype of the timestamp column
master_df.timestamp.dtype
```

```
datetime64[ns, UTC]
```

CLEAN (3) : Tidiness(1)

Define

- Merge the four columns, doggo,floofer,pupper and puppo into 1 column named dog_phases, remove the "none" values, set datatype as category.

Code

```
# Merge the four columns into a  single column with name dog_phases
master_df['dog_phases'] = master_df['doggo'] + master_df['floofer'] + master_df['pupper'] + master_df['puppo']
```

```
# Make the dog_phases a category datatype
master_df['dog_phases'] = master_df['dog_phases'].astype('category')
```

```
#change "None" text in dog stages to NaN
master_df.doggo.replace('None', '', inplace = True)
master_df.floofer.replace('None', '',inplace = True)
master_df.puppo.replace('None', '', inplace = True)
master_df.pupper.replace('None', '',inplace = True)
```

```
# drop floofer,puppo, doggo and puper column with inplace=True to make it permanent
master_df.drop(['floofer', 'puppo', 'doggo', 'pupper'], axis = 1, inplace = True)
```

Test

```
# reveal the datatype of dog_phases and entries within the dog_phases column
master_df.dog_phases.dtypes
```

```
CategoricalDtype(categories=['NoneNoneNoneNone', 'NoneNoneNonepuppo',
                  'NoneNonepupperNone', 'NoneflooferNoneNone',
                  'doggoNoneNoneNone', 'doggoNoneNonepuppo',
                  'doggoNonepupperNone', 'doggoflooferNoneNone'],
, ordered=False)
```

- There is a need to remove the multiple 'none' entries that arose as a result of the combination of the rows

```
# remove the 'none' from the categories in order to have real dog developmental stages (dog_phases)
master_df['dog_phases'] = master_df['dog_phases'].map(lambda x: x.replace("None",""))
master_df['dog_phases'].value_counts()
```

```
               1753
pupper          210
doggo            67
puppo            23
doggopupper      11
floofer           7
doggopuppo        1
doggofloofer      1
Name: dog_phases, dtype: int64
```

CLEAN (4) : Quality(8)

Define

- Convert the tweet_id datatype from integer to string string.

Code

```
# Convert tweet_id to string datatype
master_df['tweet_id'] = master_df['tweet_id'].astype('string')
```

Test

```
# Check the datatype of tweet_id column
master_df.tweet_id.dtypes
```

```
string[python]
```

CLEAN (5) : Quality (5)

Define

- Remove retweets (reply to tweets-RT)

Code

```
# Rows with retweets/reply that shows capital letter RT as the first 2 alphabets in the text column
text_RT = master_df.loc[master_df['text'].str.startswith('RT') == True]
```

```
# Display rows with retweets
retweets = master_df[master_df['retweeted_status_id'].notnull()]
```

```
# Display rows with 'NaN' in retweet_id column. Rows without retweet
retweet_null = master_df[master_df['retweeted_status_id'].isnull()]
```

```
# remove retweet rows
master_df = master_df [master_df ['retweeted_status_id'].isnull()]
```

Test

```
# check the rows and columns after the retweets has been removed.
master_df.shape
```

```
(1994, 27)
```

There are 1994 rows without retweets.

CLEAN (6): Quality (6)

Define

- Remove irrelevant columns from the master table, these are

```
# Remove irrelevant columns

master_df.drop(['in_reply_to_status_id',
                'in_reply_to_user_id',
                'retweeted_status_id',
                'retweeted_status_user_id',
                'retweeted_status_timestamp'], axis=1, inplace=True)
```

CLEAN(7): Quality 2

Define

- Extract the url from the text, place them in another column

Code

```
#Create a new column and Use regex to extract tweet's link from text
master_df['tweet_link'] = master_df.text.str.extract("(https://t.co/[a-zA-Z0-9-]+)", expand = True)
```

Storing Data

```
master_df.to_csv('twitter_archive_master.csv', index = False)
```

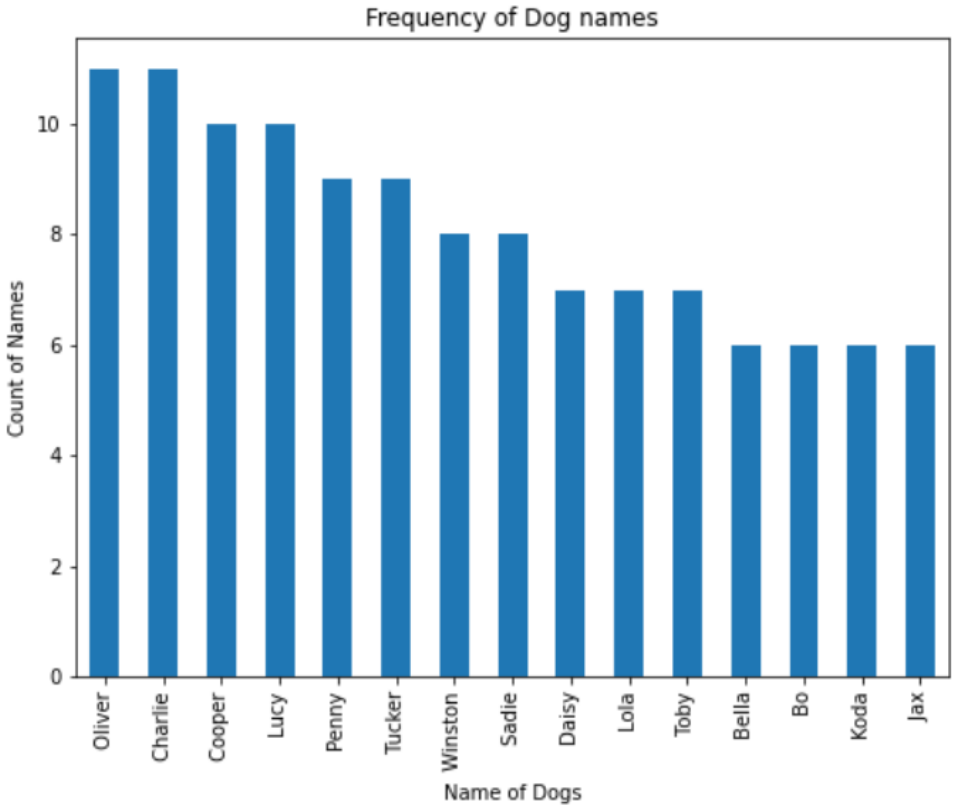
Analyzing and Visualizing data

In this section I will answer the following questions:

- What are most common dogs' names?
- What can we say about dogs' ratings?
- Why is floofer the least preferred dog developmental stage?

```
names = master_df['dog_names'].value_counts().head(15)
```

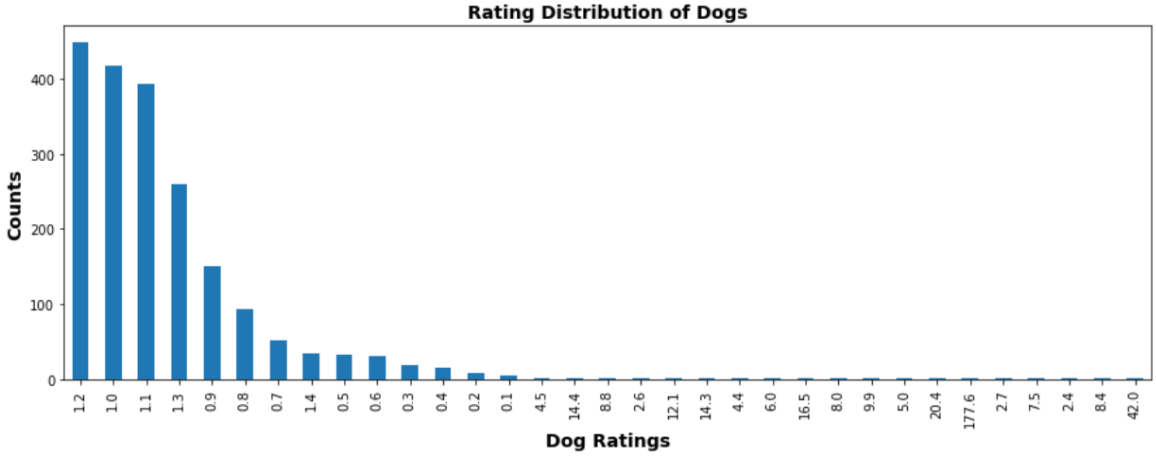
```
# Plot a bar chart of the common dog names
names.plot(kind='bar', title ='Frequency of Dog names', xlabel= 'Name of Dogs', ylabel='Count of Names', figsize=(8,6));
```



This chart shows that people prefare names like Oliver, Charlie, Copperand Lucy for there dogs.

What can we say about dog ratings?

```
# This barchart will display  distribution of tdog ratings
master_df['new_rating'].value_counts().plot(kind = 'bar', figsize=(15,5));
plt.xlabel("Dog Ratings", fontsize = 14, fontweight = 'bold')
plt.ylabel("Counts", fontsize = 14, fontweight = 'bold');
plt.title('Rating Distribution of Dogs', fontsize = 14, fontweight = 'bold');
```

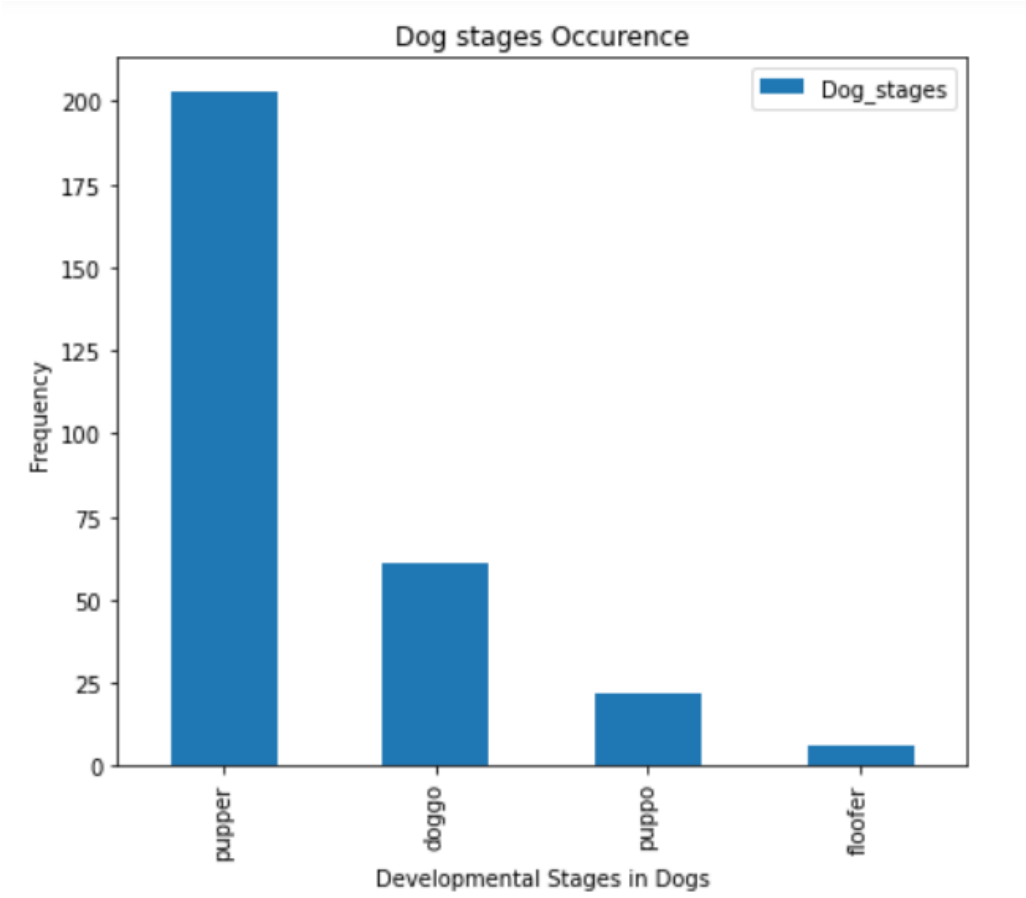


Most people rated there dogs 12/10, followed by 10/10 and 11/10.

```
plotdata = pd.DataFrame({"Dog_stages": [203, 61, 22, 6]}, index=["pupper", "doggo", "puppo", "floofer"])
# Plot a bar chart
plotdata.plot(kind="bar", xlabel= 'Developmental Stages in Dogs', ylabel = 'Frequency',
              title = 'Dog stages Occurence', figsize = (7,6));
```

Why is floofer the least prefered dog developmental stage?

```
plotdata = pd.DataFrame({"Dog_stages": [203, 61, 22, 6]}, index=["pupper", "doggo", "puppo", "floofer"])
# Plot a bar chart
plotdata.plot(kind="bar", xlabel= 'Developmental Stages in Dogs', ylabel = 'Frequency',
              title = 'Dog stages Occurence', figsize = (7,6));
```



From the dictionary of dogs, floof are dogs with seemingly excess fur. In the bar chart above, floofers are the least preferred amongst the dog stages. Could the excess fur be the reason?