wrangle_act

November 3, 2020

1 WeRateDogs Twitter Feed

This project looks at various data sources for Tweets from the WeRateDogs Twitter account, specifically:

1. the `twitter-archive-enhanced.csv` which contains the tweet text, as is the core data set
2. the Twitter API is used to access the original tweets to retrieve missing fields such as the retweet and favorite counts
3. an image prediction file containing the top 3 predictions for each of the (up to 4) dog pictures in the tweet

Having gathered the data, we assess, clean and analyse it.

1.1 Gather

We use a number of data assets including remote files on web servers, and JSON payloads returned by the Twitter API.

1.1.1 Gather the enhanced Tweets data

Pandas `read_csv()` function is quite versatile when uploading data, and can be configured to handle different date formats, numeric data types, not available (NA) markers, etc. Getting this right upfront can save time, but requires the raw data in files to be eyeballed first. For this we can use command line tools like head & tail, or alternatively Excel, which allows column headings to be frozen, data to be sorted and searched, etc.

Having looked at the raw data, we make the following observations:

1. tweet Ids are large integers, we need to select an approriate integer datatype so no accuracy is lost
2. some tweet Ids use floats, e.g.: `in_reply_to_status_id`, `in_reply_to_user_id`, with NaNs used as a Not Available marker, as mentioned above these need to be converted to integers
3. time stamps are close to ISO 8601 format, and are GMT

Actions taken to address above observations:

- convert floating point tweets Ids to a 64-bit integer, retaining the Not Available representation
- specifcally tell Pandas which columns are dates

Load the enhanced Twitter archive, using explicit data types for fields, instead of letting Pandas infer them. The Twitter API will define the data types for the Twitter sourced fields.

To get around the fact that nullable numeric fields are interpreted by `read_csv()` as floats (thus allowing NaNs to represent null), we will map nullable tweet Ids to the Pandas nullable integer data type (Int64).

```
(2356, 16)
```

The first discrepancy we note is that, according to the project motivation document, the main "archive contains basic tweet data for all 5000+ of their tweets" however that is clearly not the case as, having loaded it, the number of tweets is less than half that. As this is the master data set we have been provided with, this is the data we have to go with, since it has been previously enhanced.

To sanity check this row count, and make sure we have actually read in all the eprovided data, we will run a line count on the input file, which should roughly match the number of rows in the data frame. Any discrepancy on counts is due to those embeded new line (NL) characters in the tweet text, since the number of NL characters is what `wc` bases its line counts on.

```
2518 data/twitter-archive-enhanced.csv
```

Now we can double check the column data types, against the data type mapping provided to `read_csv()`.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2356 entries, 892420643555336193 to 666020888022790149
Data columns (total 16 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   in_reply_to_status_id       78 non-null     Int64
 1   in_reply_to_user_id         78 non-null     Int64
 2   timestamp                   2356 non-null   datetime64[ns, UTC]
 3   source                      2356 non-null   object
 4   text                        2356 non-null   string
 5   retweeted_status_id         181 non-null    Int64
 6   retweeted_status_user_id    181 non-null    Int64
 7   retweeted_status_timestamp  181 non-null    datetime64[ns, UTC]
 8   expanded_urls               2297 non-null   string
 9   rating_numerator            2356 non-null   int32
 10  rating_denominator          2356 non-null   int32
 11  name                        2356 non-null   string
 12  doggo                       2356 non-null   string
 13  floofer                     2356 non-null   string
 14  pupper                      2356 non-null   string
 15  puppo                       2356 non-null   string
dtypes: Int64(4), datetime64[ns, UTC](2), int32(2), object(1), string(7)
memory usage: 303.7+ KB
```

1.1.2 Gather the Twitter API enrichment data

Next we want to use the Twitter API to retrieve the original tweets, so that we can enrich our enhanced tweets data with the missing attributes previously idientified (`retweet_counts`, `favorite_counts`).

Having registered with Twitter as a developer, and obtained credentials and keys, we stored these in a private project directory and configuration file (which are excluded from our git repo, and thus won't be visible online in github).

We now use those credentials to authenticate with Twitter for API access.

Next we will load the enrichment data in batches, for better performance, as API invocations are subject to significant network latency. Twitter also applies rate limiting to their APIs, so it is necessary to throttle the rate at which we make requests, and to retry any failed requests. Luckily, this can be handled automatically by the Tweepy library, by setting the `wait_on_rate_limit_notify` flag when configuring API connection.

```
CPU times: user 2.1 s, sys: 114 ms, total: 2.21 s
Wall time: 13.1 s
```

```
(2331, 2)
```

Again, we briefly double check on the expected column data type mapping.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2331 entries, 666020888022790149 to 892420643555336193
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   retweet_counts   2331 non-null   Int32
 1   favorite_counts  2331 non-null   Int32
dtypes: Int32(2)
memory usage: 41.0 KB
```

1.1.3 Gather the breed prediction data

Finally we need to load the breed prediction data. We will read this data from the CloudFront URL, as opposed to the local filesystem, to ensure we get the most up-to-date version.

```
(2075, 11)
```

And finally we check for correct data type mapping.

```
<class 'pandas.core.frame.DataFrame'>
Index: 2075 entries, 666020888022790149 to 892420643555336193
Data columns (total 11 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   jpg_url  2075 non-null   string
```

```
 1   img_num  2075 non-null   int32
 2   p1       2075 non-null   string
 3   p1_conf  2075 non-null   float32
 4   p1_dog   2075 non-null   bool
 5   p2       2075 non-null   string
 6   p2_conf  2075 non-null   float32
 7   p2_dog   2075 non-null   bool
 8   p3       2075 non-null   string
 9   p3_conf  2075 non-null   float32
 10  p3_dog   2075 non-null   bool
dtypes: bool(3), float32(3), int32(1), string(4)
memory usage: 119.6+ KB
```

1.2 Assess

Having gathered the data we will now assess it, ideally both visually and programmatically.

Some of this visual assesment has already been done against the raw data in files, to ensure we used appropriate data types when uploading the data. Therefore some data quality issues (large integers stored as floating point, with potential loss of accuracy, which invalidates their meaning as an identifier) have been addressed at upload time.

1.2.1 Visual assessment

We will inspect the data that has been uploaded into the corresponding dataframes.

Enhanced tweets We assess some tweets that include a dog stage name.

```
                    in_reply_to_status_id  in_reply_to_user_id  \
tweet_id
890240255349198849                   <NA>                 <NA>
889665388333682689                   <NA>                 <NA>
889531135344209921                   <NA>                 <NA>
886366144734445568                   <NA>                 <NA>
884162670584377345                   <NA>                 <NA>

                                   timestamp  \
tweet_id
890240255349198849 2017-07-26 15:59:51+00:00
889665388333682689 2017-07-25 01:55:32+00:00
889531135344209921 2017-07-24 17:02:04+00:00
886366144734445568 2017-07-15 23:25:31+00:00
884162670584377345 2017-07-09 21:29:42+00:00

                                              source  \
tweet_id
```

```
890240255349198849  <a href="http://twitter.com/download/iphone"
rel="nofollow">Twitter for iPhone</a>
889665388333682689  <a href="http://twitter.com/download/iphone"
rel="nofollow">Twitter for iPhone</a>
889531135344209921  <a href="http://twitter.com/download/iphone"
rel="nofollow">Twitter for iPhone</a>
886366144734445568  <a href="http://twitter.com/download/iphone"
rel="nofollow">Twitter for iPhone</a>
884162670584377345  <a href="http://twitter.com/download/iphone"
rel="nofollow">Twitter for iPhone</a>

                    text  \
tweet_id
890240255349198849         This is Cassie. She is a college pup. Studying
international doggo communication and stick theory. 14/10 so elegant much
sophisticate https://t.co/t1bfwz5S2A
889665388333682689                                    Here's a puppo that seems
to be on the fence about something haha no but seriously someone help her. 13/10
https://t.co/BxvuXk0UCm
889531135344209921                          This is Stuart. He's sporting his
favorite fanny pack. Secretly filled with bones only. 13/10 puppared puppo
#BarkWeek https://t.co/y70o6h3isq
886366144734445568           This is Roscoe. Another pupper fallen victim to
spontaneous tongue ejections. Get the BlepiPen immediate. 12/10 deep breaths
Roscoe https://t.co/RGE08MIJox
884162670584377345  Meet Yogi. He doesn't have any important dog meetings today
he just enjoys looking his best at all times. 12/10 for dangerously dapper doggo
https://t.co/YSI00BzTBZ

                    retweeted_status_id  retweeted_status_user_id  \
tweet_id
890240255349198849                <NA>                      <NA>
889665388333682689                <NA>                      <NA>
889531135344209921                <NA>                      <NA>
886366144734445568                <NA>                      <NA>
884162670584377345                <NA>                      <NA>

                   retweeted_status_timestamp  \
tweet_id
890240255349198849                        NaT
889665388333682689                        NaT
889531135344209921                        NaT
886366144734445568                        NaT
884162670584377345                        NaT

                                                          expanded_urls  \
tweet_id
```

```
890240255349198849
https://twitter.com/dog_rates/status/890240255349198849/photo/1
889665388333682689
https://twitter.com/dog_rates/status/889665388333682689/photo/1
889531135344209921
https://twitter.com/dog_rates/status/889531135344209921/photo/1
886366144734445568  https://twitter.com/dog_rates/status/886366144734445568/phot
o/1,https://twitter.com/dog_rates/status/886366144734445568/photo/1
884162670584377345
https://twitter.com/dog_rates/status/884162670584377345/photo/1
```

```
                    rating_numerator  rating_denominator    name  doggo  \
tweet_id
890240255349198849                14                  10  Cassie  doggo
889665388333682689                13                  10    None   None
889531135344209921                13                  10  Stuart   None
886366144734445568                12                  10  Roscoe   None
884162670584377345                12                  10    Yogi  doggo

                   floofer  pupper  puppo
tweet_id
890240255349198849    None    None   None
889665388333682689    None    None  puppo
889531135344209921    None    None  puppo
886366144734445568    None  pupper   None
884162670584377345    None    None   None
```

We observe the following:

1. HTML in the `source` columns, with a lot of repetition (to be verified programmatically)
2. the varios rewteet columns frequently hold null values
3. on occasions multiple values appearing in the `expanded_urls` column, including repeating values
4. quite often no dog stage can be identified, and occasionally no dog name
5. dog stages place the stage name in a column named after the stage, this is redundant information

Retweet and favorite counts

```
                    retweet_counts  favorite_counts
tweet_id
666020888022790149             454             2356
666029285002620928              41              119
666033412701032449              39              109
666044226329800704             125              265
666049248165822465              40               96
```

There are no immediate issues observed by assessing a small sample of the tweet counts data

visually.

Breed predictions

```
                                                                jpg_url  img_num  \
tweet_id
666020888022790149  https://pbs.twimg.com/media/CT4udn0WwAA0aMy.jpg        1
666029285002620928  https://pbs.twimg.com/media/CT42GRgUYAA5iDo.jpg        1
666033412701032449  https://pbs.twimg.com/media/CT4521TWwAEvMyu.jpg        1
666044226329800704  https://pbs.twimg.com/media/CT5Dr8HUEAA-lEu.jpg        1
666049248165822465  https://pbs.twimg.com/media/CT5IQmsXIAAKY4A.jpg        1

                                      p1   p1_conf  p1_dog  \
tweet_id
666020888022790149  Welsh_springer_spaniel  0.465074    True
666029285002620928                redbone  0.506826    True
666033412701032449         German_shepherd  0.596461    True
666044226329800704     Rhodesian_ridgeback  0.408143    True
666049248165822465      miniature_pinscher  0.560311    True

                                   p2   p2_conf  p2_dog                  p3  \
tweet_id
666020888022790149              collie  0.156665    True    Shetland_sheepdog
666029285002620928  miniature_pinscher  0.074192    True  Rhodesian_ridgeback
666033412701032449           malinois  0.138584    True           bloodhound
666044226329800704             redbone  0.360687    True   miniature_pinscher
666049248165822465          Rottweiler  0.243682    True             Doberman

                    p3_conf  p3_dog
tweet_id
666020888022790149  0.061428    True
666029285002620928  0.072010    True
666033412701032449  0.116197    True
666044226329800704  0.222752    True
666049248165822465  0.154629    True
```

We observe the following:

1. each row refers to an image
2. each image is numbered, as it is selected as the best of up to 4 dog images that may be associated with each tweet
3. we then have the top 3 breed predictions for that image

Each prediction consists of the following information:

1. a predicted label or class (e.g.: the dog breed) that describes the image
2. a confidence estimate associated with the above prediction, in the range 0.0 -> 1.0 (0% to 100% confident)
3. a boolean indicator confirming if the predicted label is a dog breed, or some other object

Looking at the confidence estimates for predictions p1 - p3, they appear to be listed in most confident to least confident order. Therefore we will use the column name numeric suffix to generate a ranking column, which we can later sort by (to preserve this decreasing confidence order).

This last attribute confirms that the image classifier used to generate these prediction was trained on a broad set of images, only a subset of which are dog images labelled with their corresponding dog breed. But on occasions the classifier may have interpreted a dog image as an object other than a dog.

1.2.2 Programmatic assessment

Programmatic assesment gives us the opportunity to validate observations, and search for anomalies, across the entire dataset. This is very difficult to do visually unless the dataset is small, both in trems of the number of rows and columns.

Enhanced tweets Assess level of repetition in the `source` column, which holds an HTML anchor node.

```
<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for
iPhone</a>     2221
<a href="http://vine.co" rel="nofollow">Vine - Make a Scene</a>
91
<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>
33
<a href="https://about.twitter.com/products/tweetdeck"
rel="nofollow">TweetDeck</a>      11
Name: source, dtype: int64
```

Looking at the above results there appear to be 4 sources corresponding to the related applications: iPhone Twitter app, Vine app, Twitter web client and TweetDeck. This data contains a lot of redundant and messy information.

Check if there are tweets where more than one dog stage is mentioned.

```
14
```

Retweet and favorite counts We will quickly validate that all counts are positive.

```
retweet_counts     True
favorite_counts    True
dtype: bool
```

We will compare the number of entries in the enriched tweets dataframe to the number of entries in the tweet counts dataframe, to see of we successfully retrieved counts for all tweets from the API. The small difference in counts suggests a small number of tweets can no longer be retrieved.

```
(2356, 2331)
```

Breed predictions We will validate the assumption made earlier that the confidence estimates are ordered by the numeric suffix of the column name, which can be used to populate a ranking.

```
True
```

Next we validate that all confidence estimates are in the range 0.0 to 1.0.

```
True
```

1.2.3 Quality issues found

As a result of the visual and programmatic assessments, the following data quality have been found, which will require data content to be cleaned.

Enhanced tweets

1. the immediate data quality concern is that the project motivation document states that the "archive contains basic tweet data for all 5000+ of their tweets" but we are loading less than half that number of tweets. **However, given the enhanced tweets dataset is our master dataset, there is nothing that we can do to remedy the much smaller number of rows, beyond highlighting this observation**
2. as previously mentioned, the issue with some tweet Id columns being treated as floating point numbers, and the fact that rounding could invalidate these, was resolved at data loading time (without impacting the fact that they are nullable columns)
3. the format of the `timestamp` is very close to an ISO 8601 timestamp, however it is missing the 't' character as the separator between the date and time portions. There are definite advantages in following a recognised standard, as this will be understood by tools such as database import utilities, however Pandas has correctly parsed dates
4. in the `source` column, extract the source app name from the HTML anchor string, and then map this column to a Pandas categorical
5. it is unclear why, in the `expanded_urls` columns, the same URL get repeated, since looking at the tweet text there is only one reference to the corresponding link. Therefore we will remove duplicates
6. convert the dog stage columns into boolean datatype, and interpret the constant value 'None' as a missing stage
7. since the dog stage column names are the stages, storing that same name as a value is redundant information, following on from the previous observation, where the dog stage appears we will just store a boolean true value

Retweet and favorite counts

1. while the intention is to obtain retweet and favorite counts for all the tweets in the enhanced tweets dataset, we cannot guarantee that the Twitter API will always return the original Tweet, e.g.: it may subsequently have been deleted
2. where the counts were successfully retrieved for the original tweet (the majority of cases, as proven in the programatic assesment), then there is a one-to-one relationship between the rows in the counts dataframe, and the rows in the enhanced tweets dataframe. Therefore the

counts columns can be merged back into the enhanced tweets dataframe, as arguably they are part of that tweet observation. In the few cases where the counts are missing, we will store nulls

Breed predictions No obvious data quality issues, beyond the prediction column names being used as variables (the numeric suffix added).

1.2.4 Structural issues

After looking at data frame structure, column naming, and inspecting values, and then applying the Tidy Data principles, the following structural issues will need to be addressed.

Enhanced tweets

1. the `source` column must store a category that represent the application (and possibly device) used to author the tweet
2. the `expanded_urls` column can store multiple values per row, depending on the web links embeded in the tweet text, therefore these observations need to be stored in a separate table (however, we will first remove any duplicate values).
3. dog stage is a multivalued categorical variable, as a tweet can reference more than one stage. Therefore we retain the existing columns but encode them in the style of one hot encoding

Retweet and favorite counts No obvious structural issues here.

Breed predictions

1. a variable (prediction number) is embeded in the column names of the prediction columns (predicted breed, prediction confidence, and is-a-dog flag)
2. the prediction number ranks the predictions in the order most confident (1st prediction) to least confident (3rd prediction)
3. the actual breed predictions should be held in a separate dataframe, and linked back to the tweet and tweet image they are associated with

1.3 Clean

We will now clean the issues uncovered during assesment using a *define/code/test* framework, which will be applied to each of the issues.

```
(2356, 16)
```

1.3.1 Extract tweet application from `source` column

Define

- parse source column which holds an HTML anchor node
- extract anchor node content, describing the application used
- convert the column to Pandas categorical, as a more efficient representation that can be used in models

Code

Test

We will check that the tweet source column is now a categorical, and the number of categories is that expected.

1.3.2 Move `expanded_urls` to a detail dataframe

Define

- split multi-valued string of comma separated URLs, into URL arrays
- remove any duplicate URLs from the array
- convert each array into list of tuples, bound to the containing `tweet_id`
- stores these tuples as rows in a new dataframe

Code

Test

We will count total and unique tweet Ids in the new dataframe holding expanded URLs. The later will be lower, accounting for multiple rows (hence web links in the tweet text) associated with the same tweet.

```
(2338, 2297)
```

1.3.3 Convert dog stage columns to boolean

Define

- where the value 'None' is stored, set False, otherwise set True

Code

Test

We will check that the dog stage columns are now boolean type.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2356 entries, 892420643555336193 to 666020888022790149
Data columns (total 15 columns):
```

```
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   in_reply_to_status_id       78 non-null     Int64
 1   in_reply_to_user_id         78 non-null     Int64
 2   timestamp                   2356 non-null   datetime64[ns, UTC]
 3   source                      2356 non-null   category
 4   text                        2356 non-null   string
 5   retweeted_status_id         181 non-null    Int64
 6   retweeted_status_user_id    181 non-null    Int64
 7   retweeted_status_timestamp  181 non-null    datetime64[ns, UTC]
 8   rating_numerator            2356 non-null   int32
 9   rating_denominator          2356 non-null   int32
 10  name                        2356 non-null   string
 11  doggo                       2356 non-null   bool
 12  floofer                     2356 non-null   bool
 13  pupper                      2356 non-null   bool
 14  puppo                       2356 non-null   bool
dtypes: Int64(4), bool(4), category(1), datetime64[ns, UTC](2), int32(2),
string(2)
memory usage: 205.0 KB
```

1.3.4 Merge retweet and favorite counts into enhanced tweets dataframe

Define

- merge retweet and favorite count columns into enhanced tweets dataframe, using a left join with nulls for missing count values

Code

Test

Validate number of rows after merge, including count of rows with null retweet or favorite

```
2356
```

```
retweet_counts     25
favorite_counts    25
dtype: int64
```

1.3.5 Melt image prediction column headers into detail dataframe

Define

- store `jpg_url` and `img_num` columns in a clean dataframe

- melt prediction 1 to 3 columns into temporary dataframes, with the prediction rank as a constant value, and the related `tweet_id`
- stack the above temporary dataframes into a predictions dataframe, with repeated `tweet_id` as the index

Code

```
(2075, 11)
```

Test

Validate dataframe column names and structure as expected.

```
<class 'pandas.core.frame.DataFrame'>
Index: 2075 entries, 666020888022790149 to 892420643555336193
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   jpg_url  2075 non-null   string
 1   img_num  2075 non-null   int32 
dtypes: int32(1), string(1)
memory usage: 40.5+ KB
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6225 entries, 666020888022790149 to 892420643555336193
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   pred_rank        6225 non-null   int64 
 1   pred_class       6225 non-null   string
 2   pred_confidence  6225 non-null   float32
 3   pred_is_dog      6225 non-null   bool  
dtypes: bool(1), float32(1), int64(1), string(1)
memory usage: 176.3 KB
```

1.4 Analyse

In this section we look at the data and analyse it to obtain some insights. Specifically, we are interested in:

1. Finding the number of tweets with a score above 10/10, versus tweets with a score under 10/10
2. Identify the tweets where more than one dog stage appears
3. Finding the number of top breed predictions from the image classifier, with a predictionconfidence below 0.5

Count number of scores above and below 10/10

```
(1451, 905)
```

Show tweets with more than one dog stage in the tweet text

```
                    text  \
tweet_id
855851453814013952        Here's a puppo participating in the #ScienceMarch.
Cleverly disguising her own doggo agenda. 13/10 would keep the planet habitable
for https://t.co/cMhq16isel
854010172552949760  At first I thought this was a shy doggo, but it's actually a
Rare Canadian Floofer Owl. Amateurs would confuse the two. 11/10 only send dogs
https://t.co/TXdT3tmuYk
817777686764523521  This is Dido. She's playing the lead role in "Pupper Stops
to Catch Snow Before Resuming Shadow Box with Dried Apple." 13/10 (IG:
didodoggo) https://t.co/m7isZrOBX7
808106460588765185                      Here we have Burke (pupper) and Dexter
(doggo). Pupper wants to be exactly like doggo. Both 12/10 would pet at same
time https://t.co/ANBpEYHaho
802265048156610565
Like doggo, like pupper version 2. Both 11/10 https://t.co/9IxWAXFqze
801115127852503040                      This is Bones. He's being haunted by
another doggo of roughly the same size. 12/10 deep breaths pupper everything's
fine https://t.co/55DqeOSJNj
785639753186217984  This is Pinot. He's a sophisticated doggo. You can tell by
the hat. Also pointier than your average pupper. Still 10/10 would pet
cautiously https://t.co/f2wmLZTPHd
781308096455073793
Pupper butt 1, Doggo 0. Both 12/10 https://t.co/WQvcPEpH2u
775898661951791106
RT @dog_rates: Like father (doggo), like son (pupper). Both 12/10
https://t.co/pG2inLaOda
770093767776997377                                                      RT
@dog_rates: This is just downright precious af. 12/10 for both pupper and doggo
https://t.co/o5J479bZUC
759793422261743616                      Meet Maggie &amp;amp; Lila. Maggie is the
doggo, Lila is the pupper. They are sisters. Both 12/10 would pet at the same
time https://t.co/MYwR4DQKll
751583847268179968                          Please stop sending it pictures
that don't even have a doggo or pupper in them. Churlish af. 5/10 neat couch tho
https://t.co/u2c9c7qSg8
741067306818797568
This is just downright precious af. 12/10 for both pupper and doggo
https://t.co/o5J479bZUC
733109485275860992
Like father (doggo), like son (pupper). Both 12/10 https://t.co/pG2inLaOda

                    doggo  floofer  pupper  puppo
tweet_id
855851453814013952   True    False   False   True
854010172552949760   True     True   False  False
817777686764523521   True    False    True  False
```

```
808106460588765185   True    False    True  False
802265048156610565   True    False    True  False
801115127852503040   True    False    True  False
785639753186217984   True    False    True  False
781308096455073793   True    False    True  False
775898661951791106   True    False    True  False
770093767776997377   True    False    True  False
759793422261743616   True    False    True  False
751583847268179968   True    False    True  False
741067306818797568   True    False    True  False
733109485275860992   True    False    True  False
```

Count tweets where the top scoring breed prediction is below 0.5

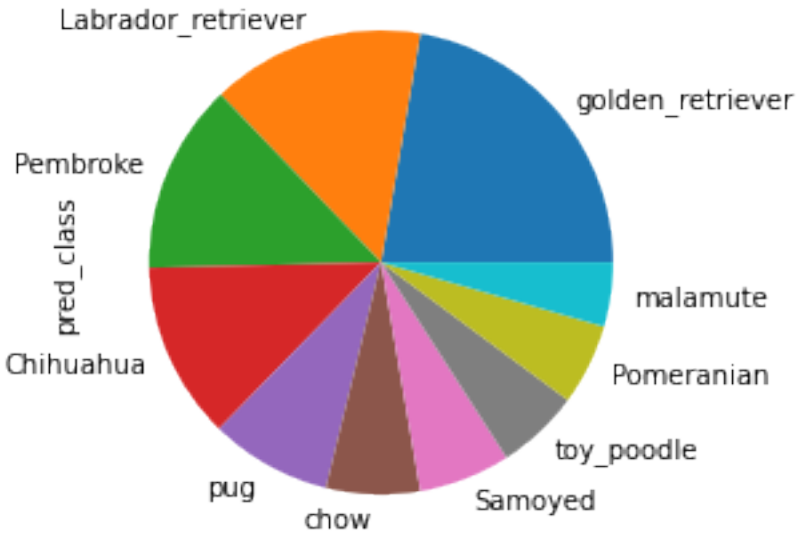
```
559
```

Now we are going to generate some visualisations:

1. First, based on the top image prediction, look at the frequency distribution for the top 10 breeds only, based on number of tweets
2. Now look at the frequency distribution for the top 10 breeds only, based on aggregate number of favorites

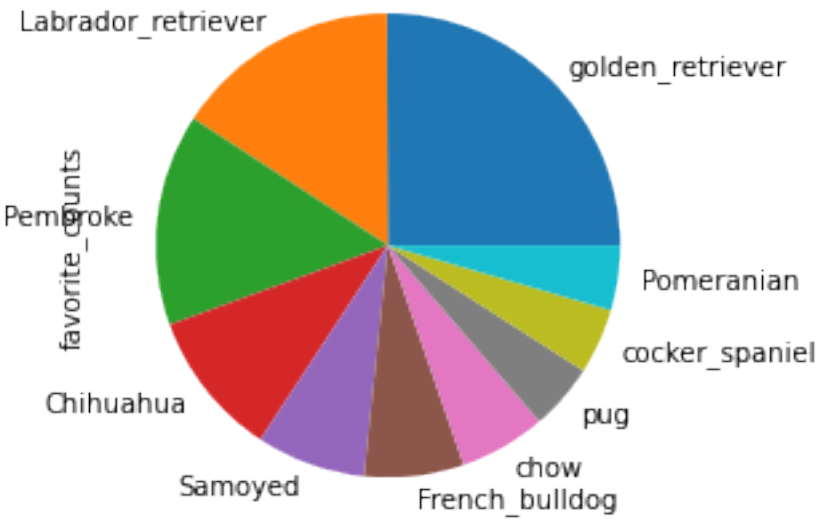
Breed prediction distribution by number of tweets

```
<AxesSubplot:ylabel='pred_class'>
```



Breed prediction distribution by number of favorites

```
<AxesSubplot:ylabel='favorite_counts'>
```



1.5 Predict breeds from images

TBD

1.5.1 Generate internal report

Having cleaned the data, and generated data insights, we can now generate the internal documentation from this notebook's markdown cells.

(you probably want to clear all output previous to the data insights output generated in the last section, and then SAVE the notebook)

```
[NbConvertApp] Converting notebook wrangle_act.ipynb to pdf
[NbConvertApp] ERROR | Error while converting 'wrangle_act.ipynb'
Traceback (most recent call last):
  File "/opt/conda/lib/python3.6/site-packages/nbconvert/nbconvertapp.py", line
410, in export_single_notebook
    output, resources = self.exporter.from_filename(notebook_filename,
resources=resources)
  File "/opt/conda/lib/python3.6/site-packages/nbconvert/exporters/exporter.py",
line 179, in from_filename
    return self.from_file(f, resources=resources, **kw)
  File "/opt/conda/lib/python3.6/site-packages/nbconvert/exporters/exporter.py",
line 197, in from_file
```

```
    return self.from_notebook_node(nbformat.read(file_stream, as_version=4),
resources=resources, **kw)
  File "/opt/conda/lib/python3.6/site-packages/nbconvert/exporters/pdf.py", line
171, in from_notebook_node
    nb, resources=resources, **kw
  File "/opt/conda/lib/python3.6/site-packages/nbconvert/exporters/latex.py",
line 88, in from_notebook_node
    return super(LatexExporter, self).from_notebook_node(nb, resources, **kw)
  File "/opt/conda/lib/python3.6/site-
packages/nbconvert/exporters/templateexporter.py", line 322, in
from_notebook_node
    output = self.template.render(nb=nb_copy, resources=resources)
  File "/opt/conda/lib/python3.6/site-packages/jinja2/environment.py", line
1090, in render
    self.environment.handle_exception()
  File "/opt/conda/lib/python3.6/site-packages/jinja2/environment.py", line 832,
in handle_exception
    reraise(*rewrite_traceback_stack(source=source))
  File "/opt/conda/lib/python3.6/site-packages/jinja2/_compat.py", line 28, in
reraise
    raise value.with_traceback(tb)
  File "/opt/conda/lib/python3.6/site-
packages/nbconvert/exporters/../templates/latex/article.tplx", line 8, in top-
level template code
    ((* extends cell_style *))
  File "/opt/conda/lib/python3.6/site-
packages/nbconvert/exporters/../templates/latex/style_jupyter.tplx", line 176,
in top-level template code
    \prompt{(((prompt)))}{(((prompt_color)))}{(((execution_count)))}{(((extra_sp
ace)))}
  File "/opt/conda/lib/python3.6/site-
packages/nbconvert/exporters/../templates/latex/base.tplx", line 7, in top-level
template code
    ((*- extends 'document_contents.tplx' -*))
  File "/opt/conda/lib/python3.6/site-
packages/nbconvert/exporters/../templates/latex/document_contents.tplx", line
50, in top-level template code
    ((*- block figure scoped -*))
  File "/opt/conda/lib/python3.6/site-
packages/nbconvert/exporters/../templates/latex/skeleton/display_priority.tplx",
line 5, in top-level template code
    ((*- extends 'null.tplx' -*))
  File "/opt/conda/lib/python3.6/site-
packages/nbconvert/exporters/../templates/latex/skeleton/null.tplx", line 30, in
top-level template code
    ((*- block body -*))
  File "/opt/conda/lib/python3.6/site-
packages/nbconvert/exporters/../templates/latex/base.tplx", line 197, in block
```

```
"body"
    ((( super() )))
  File "/opt/conda/lib/python3.6/site-
packages/nbconvert/exporters/../templates/latex/skeleton/null.tplx", line 32, in
block "body"
    ((*- block any_cell scoped -*))
  File "/opt/conda/lib/python3.6/site-
packages/nbconvert/exporters/../templates/latex/skeleton/null.tplx", line 85, in
block "any_cell"
    ((*- block markdowncell scoped-*)) ((*- endblock markdowncell -*))
  File "/opt/conda/lib/python3.6/site-
packages/nbconvert/exporters/../templates/latex/document_contents.tplx", line
67, in block "markdowncell"
    ((( cell.source | citation2latex | strip_files_prefix |
convert_pandoc('markdown+tex_math_double_backslash', 'json',extra_args=[]) |
resolve_references | convert_pandoc('json','latex'))))
  File "/opt/conda/lib/python3.6/site-packages/nbconvert/filters/pandoc.py",
line 26, in convert_pandoc
    return pandoc(source, from_format, to_format, extra_args=extra_args)
  File "/opt/conda/lib/python3.6/site-packages/nbconvert/utils/pandoc.py", line
53, in pandoc
    check_pandoc_version()
  File "/opt/conda/lib/python3.6/site-packages/nbconvert/utils/pandoc.py", line
101, in check_pandoc_version
    v = get_pandoc_version()
  File "/opt/conda/lib/python3.6/site-packages/nbconvert/utils/pandoc.py", line
78, in get_pandoc_version
    raise PandocMissing()
nbconvert.utils.pandoc.PandocMissing: Pandoc wasn't found.
Please check that pandoc is installed:
http://pandoc.org/installing.html
mv: cannot stat 'wrangle_act.pdf': No such file or directory
```