**Aggregating algorithm**

As per the project brief, we have provided a python file (aggregate\_algorithm.py) which implements the algorithm laid out in “A generalized approach for producing, quantifying, and validating citizen science data from wildlife images” (Swanson et al.) in order to create aggregated classifications for the photos taken as part of the MammalWeb project.

Aggregate\_algorithm uses the pymysql library to handle the connection with the SQL database in which all the photo and classification data is stored. The function aggregate\_classifications takes a photo\_id and a pymysql connection object as parameters and generates the aggregated classification along with the accuracy metrics as explained in Swanson et al.. Everything calculated is then written into an ‘aggregate’ table in the database that we have created.

The evenness, (fraction) support and (fraction) blanks attributes of the table measure the likelihood that the species given by the aggregate is correct. The age and gender are more simply the mode of the individual classifications.

The attribute ‘flag’ is a marker to show what state the aggregate for that photo has reached. These are laid out in Swanson et al. and have been added to the options table with their struc attribute as ‘flag’. Photos with a flag of ‘complete’ or ‘consensus’ should be removed from the pool of photos to be classified and the aggregate considered correct. Photos with a flag of ‘blank’ should also be removed as they almost definitely do not contain any animal. Photos with a flag of ‘incomplete’ should remain in the rotation as they require more classifications to be sure of the aggregates accuracy. Finally, if any flags are set to -1 an error has occurred and needs to be corrected.

Along with the aggregate\_classification function, the script includes a simple loop to run the algorithm on every photo in the database to completely populate the aggregate table.