Pet Recommender System Model

1. Heroku app
   1. Recommender.html (one page of the overall project presentation)
      1. Contains prose, images, and a video to present the pet recommender system
   2. Recommender.html is rendered by the Flask app
2. Locally built recommender system
   1. recommender.html works with recommender.js to render:
      1. Pet photo
      2. Pet metadata
      3. Interactive thumbs up/down buttons
      4. Instructions to the user
   2. recommender.js receives user choice from recommender.html and passes it to recommender.py
   3. recommender\_preprocess.py preprocesses the data
      1. Import pet data and convert to json string (or whatever format is needed)
         1. pets = mongo.db.pet\_data.find(only first 500 for debugging)
         2. return json\_util.dumps(pets)
      2. Pre-process data in Pandas
         1. Remove stop words
         2. Convert NaN’s to ““
         3. Convert to lower case and strip white spaces
      3. Create metadata soup with createsoup()
      4. Use CountVectorizer()
      5. Use Scikitlearn linear\_kernel() to compute Cosine similarity scores between each pet and every other pet (could use Manhattan, Euclidean, Pearson
      6. Use Pickling to store pre-processed data for use by recommender.py
   4. recommender.py runs the ML model to select the next pet to display
      1. Import CountVectorizer, PyMongo, Pandas, jsonUtil, and Numpy modules
      2. Declare arrays to hold pets in different categories
         1. not\_shown\_yet\_array = []
         2. shown\_array = [] is an array containing two arrays
            1. voted\_up = []
            2. voted\_down = []
      3. Declare update\_buckets() function
         1. Get latest user choice
         2. Move pet from
      4. Declare get\_recommendations() function

def get\_recommendations(pet\_id, cosine\_sim):

# Get the index of the pet that matches the pet\_id

idx = indices[pet\_id]

# Get the pairwise similarity scores

sim\_scores = list(enumerate(cosine\_sim[idx]))

# Sort the pets based on their similarity scores

sim\_scores = sorted(sim\_scores, key=lambda x: x[1],

reverse = True)

# Get the ID of the top-matching pet

sim\_scores = sim\_scores[1]

next\_pet\_index = sim\_scores[0]

return metadata[‘pet\_id’].iloc[next\_pet\_index]

* + 1. Run get\_recommendations() function