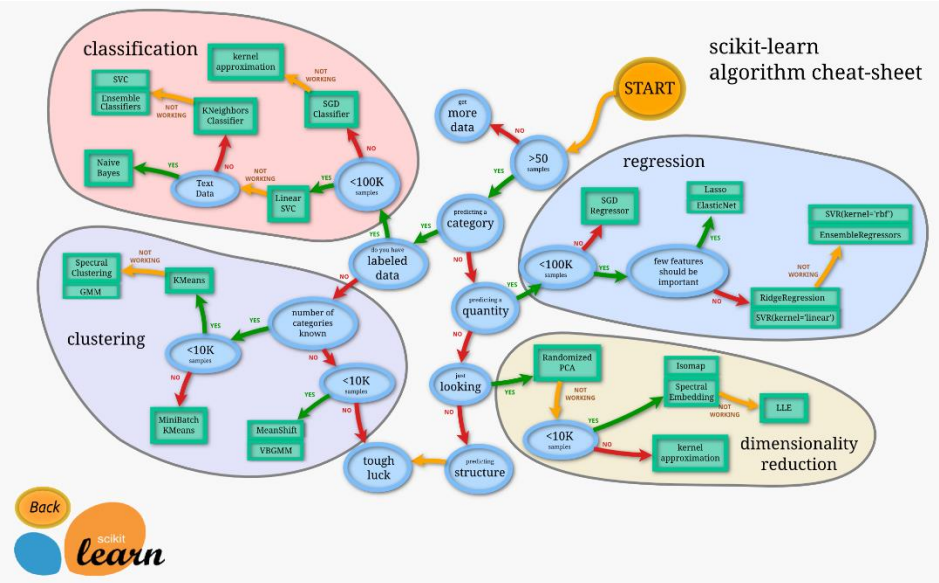
Social Computing Homework 3 Report

CODE LINK ON GOOGLE COLAB:
https://colab.research.google.com/drive/1_PtQuMQgN_5QvhR0-ytZf7xMpl4QEMds

1. Features:
    a. Since in the end, I am to use this classifier on the datasets given in the previous homework, I thought to only consider data that are present in both, so things like name, content of the messages. I discovered this article (https://www.figure-eight.com/using-machine-learning-to-predict-gender/) about the gender classification and found out that their approach was to put the lexicon of the entire dataset as their features and train on finding the words used that most likely to indicate gender. Since they have already done it with that, I thought to try and explore other features I could extract and see if I could find other features that could predict gender. The features I ended up looking at were number of male and female names found in the username. I figure that it might be common for users to put their own names into their usernames, so it might be a good indicator of gender. For both genders, I found the US census list of the 100 most popular names and checked whether or not they appeared in the username. Then I looked at gendered nouns in the text and descriptions and counted how many male and female nouns appeared. These might indicate, especially in the description, the gender of the user as they might use these words to describe themselves. As for the text, I looked at the sentiment and Parts of Speech counts for it. They were relatively simple to extract and I was curious whether they would have any bearing on gender. I used built in sentiment and POS taggers in the nltk python library.
2. Classifier
    a. I used LinearSVC because it was recommended by this chart on the scikit website:



I also did attempt to use Naïve Bayes and K nearest neighbors, but they performed worse than SVC, though to be honest, that isn't really saying much.

3. Evaluation
    a. I ran k-fold validation so I could kind of get a consistent result for the training and I was able to generate graphs for accuracy and recall with the built in libraries.
4. Implementation:
    a. I first imported and cleaned the data using pandas. I removed the irrelevant data that I did not want to look at. This included things like the profile picture url, color, and so on. Then I also dropped all the rows containing genders that were either unknown or brand gender. Then I converted the gender into binary, 1 for male and 0 for female. This ended up reducing the data to about 13000 values. Then I ran my data preprocessing. I used a combination of scikit learn and nltk tools to tokenize and clean the data of punctuation. Then for the male and female nouns, I looked at the descriptions see if they contained any words from the 2 lists composed of common male and female nouns from here: https://www.onlinemathlearning.com/gender-nouns.html. I kept track of their counts and used that as the feature. Then I looked at the usernames and counted whether or not they contained a male or female name. I check the usernames with a list of the top 100 most common US male and female names. Then I looked at the text of the tweet. I first cleaned it a bit more by removing links. The most common links seen are other twitter with very few other websites. I removed them and then tokenized them. With the tokenized text I extracted sentiment using the built in vader lexicon from nltk and extracted pos tags using a built in nltk module. I stored the values of the sentiment (gave me numbers for positive, negative, neutral and compound) and stored the number of the most common POS that appeared. For my model, I used scikit learn and used the provided LinearSVC module. I just had to split my data to the features and target, which was the gender column. Then I ran k folds validation 7 times with the SVC model and got my metrics using the built in cross validate function in scikit. I had to just pass in the data, the target, how many times to run the learning, and what scores I wanted and it would output a dictionary of results.

5. Analysis
    a. My model had very poor performance It performed not much better that random guessing, which would be a baseline of around 50%, my model's best performance was at a peak of 55%. The following is a table of the results:

KTH ITERATIONS	ACCURACY	PRECISION	RECALL
1	0.48682171	0.48238255	0.92741935
2	0.48449612	0.48169243	0.95483871
3	0.55658915	0.56521739	0.33548387
4	0.52093023	0.66666667	0.00645161
5	0.48642358	0.48286853	0.97899838
6	0.4996121	0.48828829	0.87560582
7	0.48719938	0.48164336	0.8901454
8	0.55314197	0.58431373	0.24071082
9	0.52831652	0.51957295	0.2358643
10	0.52288596	0.55263158	0.03392569
AVERAGE	0.512641672	0.530527748	0.547944395

    If better features were used, as in the linked article, such as taking what kind of words were in each tweet into account, my model would have performed better.
6. Conversational data
    a. I had to adjust the conversational data slightly to sort of match the classifier by setting some features to zero. The classifier performed not much better on the conversational data sets, getting an average accuracy of 0.5126463700234193, an average precision of 0.2330456391365554 and an average recall of 0.07564766839378238on Mar06GroupA and 0.52576112412178 average accuracy, 0.2439570330525509 average precision, and 0.04870466321243523 average recall for Mar07GroupB over ten trials.
7. Conclusion
    a. While the model I made was garbage, I did learn a lot more about how to use ML and nlp tools and I felt I learned quite a bit. Hopefully I'll get much better results on hw3!

CODE LINK ON GOOGLE COLAB:
https://colab.research.google.com/drive/1_PtQuMQgN_5QvhR0-ytZf7xMpl4QEMds