**1. What areas of machine learning models are you most familiar with?**

I am most familiar with Supervising Machine learning regarding classification models such as:

* Random Forest
* Support Vector Machines
* Naïve Bayes
* Adaptive Boosting Classifier

I am also familiar with regression models such as:

- OLS

* Lasso
* Ridge
* Elastic Net

I am familiar with Clustering models such as:

* K means
* K-Medoids

**2. Imagine you are designing a system for Github, which needs to learn as much about each engineer as possible in order to provide search and recommendations to recruiters looking for particular sorts of engineers.**

**What dataset(s) would you build?**

I would scrape Github profiles for Username, introductions, names and descriptions of projects, names and descriptions of repositories, information from readme files, and contribution activity. I would then save each person’s information in a database that I can access later for modeling. Names of projects and activities would indicate what type of engineer the recruiter is looking for. The metadata would be removed, only keeping descriptive information for modeling.

**What features would you use?**

I would use the text itself from each profile as the features. I would have a column for name of profile, the introduction, projects/descriptions, contributions, and readme files. I would remove Unicode, stem, and tokenize the words. These tokenized words would be vectorized to apply machine-learning models to it later on. All the descriptive text for each user would initially be separate, but it would be combined for the modeling portion. Github stars would be tracked as a feature. More stars would indicate more people are following that user, and that person’s work is probably interesting or helpful. It would not be use for modeling, but it would be a good descriptive statistic to display to the recruiter as an indication of a strong engineer.

**What kind(s) of ML models would you think about applying and why?**

I would apply a Doc2Vec Model. Text of each profile can be considered a “document”. This model puts the vectorized text into a high dimensional space where similar profiles will be next to each other.

**What do you think the low hanging fruit would be? (i.e. What would you expect to be the easy parts of the project with high likelihoods of good results?)**

The easiest parts of the project would be feeding the data through the Doc2Vec model. But the challenge would be to clean the data and prepping the data so a model could be applied. With web scraped data, there is a lot metadata and Unicode that needs to be removed in order for the model to work. There is also a chance that there is not enough information from some profiles to even create any similarity.

After the data is fed through the model, you get the vocabulary list from it. A recruiter can use the vocabulary list and type in keywords from it. Then, the model will return the most similar profiles in reference to the keywords used.

For example, if someone typed, “Deep Learning”, “Python”, “Tensorflow”. The most relevant profiles would come up with people who have Deep Learning projects that used Python and Tensorflow. Along with it, it would have Github ID, stars, projects and descriptions, and other relevant information for the recruiter to make a decision.

**What tools, libraries, and platforms, if any, would you think about using for this task?**

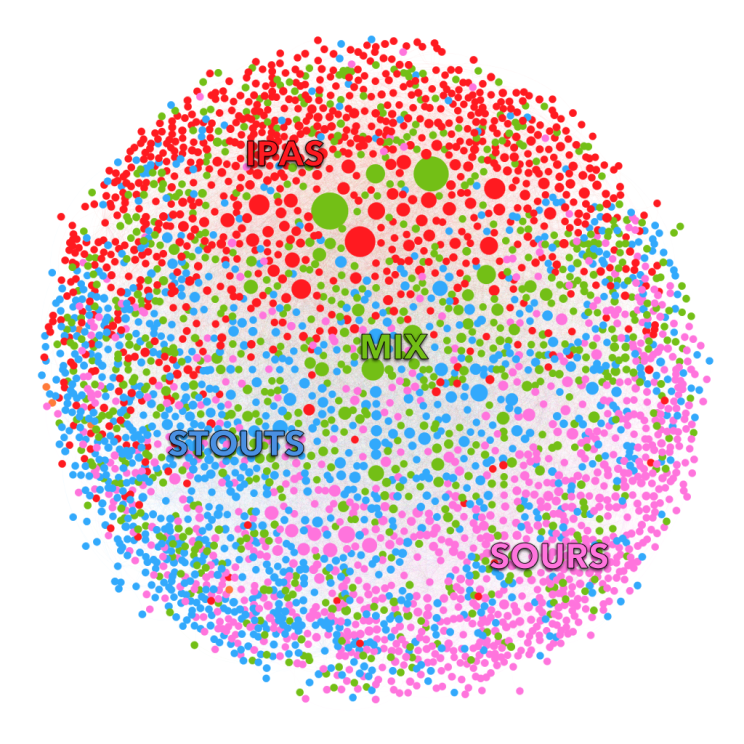
For text processing, I would use NLTK, but lately Spacy has been a powerful emerging tool for Natural Language Processing. I would use some type of web scraper in python such as Beautiful Soup to pull the data from Github, store the data in a NOSQL database that I can use later. The modeling portion would be done with Gensim, to help find most similar profiles, given keywords of interest.

To productionaliize this, I would create a flask application so recruiters can interact with the website and the most similar profiles can be displayed to the user.

As an ongoing basis, I would scrape Github for new profiles and updates. I can schedule something in Amazon Web Services where the model updates on a weekly or a daily basis. The update would run offline and then when it is done, it will automatically replace the current model without interfering with the user experience.

Textacy, which is built from the open-source package, can be used to create topic modeling to figure out which profiles group together.

This an example of beer descriptions:



As you can see stouts, sours and IPAs group together. The same idea can be applied to Github profiles, where similar profiles will start grouping together.