How well can one predict wine type using product description alone? In this post, I will attempt to answer this question, which requires scraping the web, processing some natural language, and training a random forest classification model. I'll complete all of these tasks using Python. The code that implements the analysis in this post is located [here.](https://github.com/volk0ff/predict_wine_types)

Gathering Data

To answer this question, one first needs a data set that has product descriptions and types. To scrape data, I used the python[**requests**](http://docs.python-requests.org/en/master/)library, which makes creating HTTP requests and HTTP responses very easy. Using a few lines of code set inside of a for loop, one is able to download several thousand html pages containing wine type, description, and various other product-related data. To scrape each of these several thousand html documents and pick out the specific text that I need for the data set - that is, the product name, description, and type - I used the **[pyquery](https://pythonhosted.org/pyquery/" \t "_blank)** library. **Pyquery** is essentially a python implementation of Jquery; it allows the user to traverse a html document using element, class, and id selectors.

At this point, it would be useful to say a little about the data I am collecting. There are three main fields that I am gathering: name, production description, and type. "Name" is, of course, the name of the wine. "Product Description" is several sentence review of the wine, for example, the product description for the Kendall Jackson Chardonnay is:

"California- Tasty tropical flavors such as mango, papaya and pineapple with citrus notes, delicately intertwine with aromas of green apple and pear to create depth and balance throughout. A hint of toasted oak rounds out the finish."

Lastly, "type" is the class of the wine. There are eight types:

* White Wine,
* Champagne/Sparking Wine,
* Red Wine,
* Dessert & Fortified Wine,
* Rose/Blush Wine,
* Other Wine,
* Non-Alcoholic Wine, and
* Sake & Plum Wine.

After scraping all the necessary data from the site, I created a [**pandas**](http://pandas.pydata.org/) dataframe - which is essentially fancy table-like data structure - to hold the data.

Cleaning Data

To train the random forest, I needed to generate a numeric representation of each product description. To do this, I followed the [**bag of words**](http://en.wikipedia.org/wiki/Bag-of-words_model)approach. This approach creates a vocabulary from all the product descriptions then represents each description by a vector of counts for each word in the vocabulary.

Though before I could generate these word count vectors, I had to first clean the product descriptions, which required removing all non-letter characters, unnecessary spaces, and lowercasing all the words in the descriptions. In **pandas**, this process is simple by using the built-in string methods. Then I removed all the words contained in the wine types field from the descriptions. I did not want to have the wine type explicitly stated in the description. That would make classification too easy.

Training a Random Forest

After cleaning the data and creating a numeric representation each product description, I then trained the random forest. So what is a random forest? It is a type of machine learning algorithm that can be used to classify observations into discrete categories. In this case, the discrete categories are types of wine. The algorithm works by constructing a large number of decision trees based upon a set of independent variables (in this case, the component words of product description), making a prediction of the dependent variable (in this case, wine type) with each of these trees, and then averaging the results across all trees to generate an overall prediction.

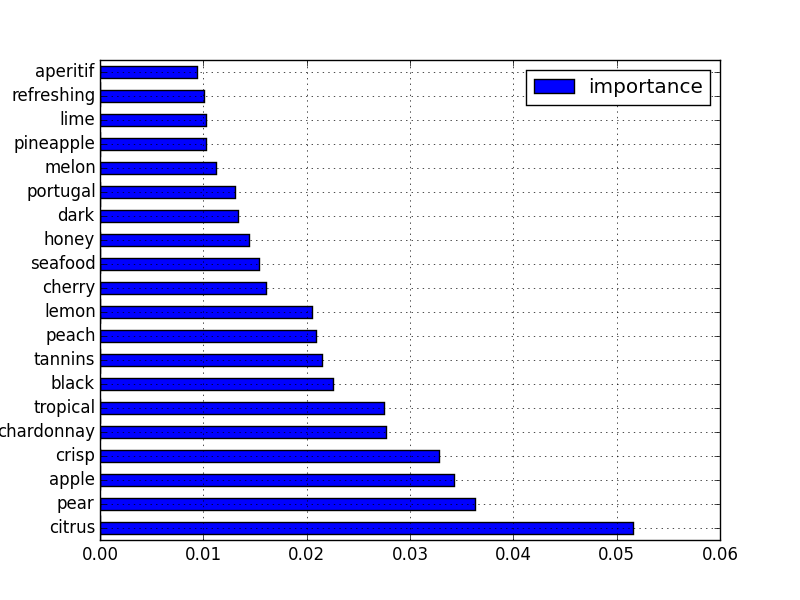
Using the vectorized product descriptions created in the previous section, training a random forest with python's **[scikit-learn](http://scikit-learn.org/stable/" \t "_blank)** library is trivial and only requires two lines of code.  For more information on random forests, see the [scikit-learn documentation](http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html" \t "_blank) or the algorithm's creator's [site](https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm).

Something interesting about a random forest is that it is not strictly necessary to run cross validation on a trained model. When a random forest is trained, each tree in the forest is trained on only about 2/3's of the data. The data that was not used in training a specific tree can be used to generate an out of sample error estimate. If you average the error estimates across all trees, you get an error estimate for the entire model. This error estimate is called an "out of bag" error estimate; one minus this error estimate is the "out of bag" score.

Results

How well did the trained random forest perform? That is, how well did the algorithm predict wine type based upon product description alone? The "out of bag" score, which is a measure of accuracy, for the trained random forest was 0.929, that is, the algorithm correctly classifies 92.9% of observations.

Another interesting question one can answer with a random forest is, what are the most important features for classifying wines into types? "Features" in this case would be specific words used in the product description. "Most important" means most effectively splits the wines into types. Below I graph the 20 most important features. The more important a feature is, the higher is its score. So "citrus" is the most useful word in classifying wines into types.



Fruit names seem to dominant the list. I don't like the fact that "chardonnay" shows up in the list. If I were to continue refining this analysis, I would clean the product descriptions even more, removing all varietals and maybe even country names. Once again, I don't want to make the classification problem too easy for the random forest.