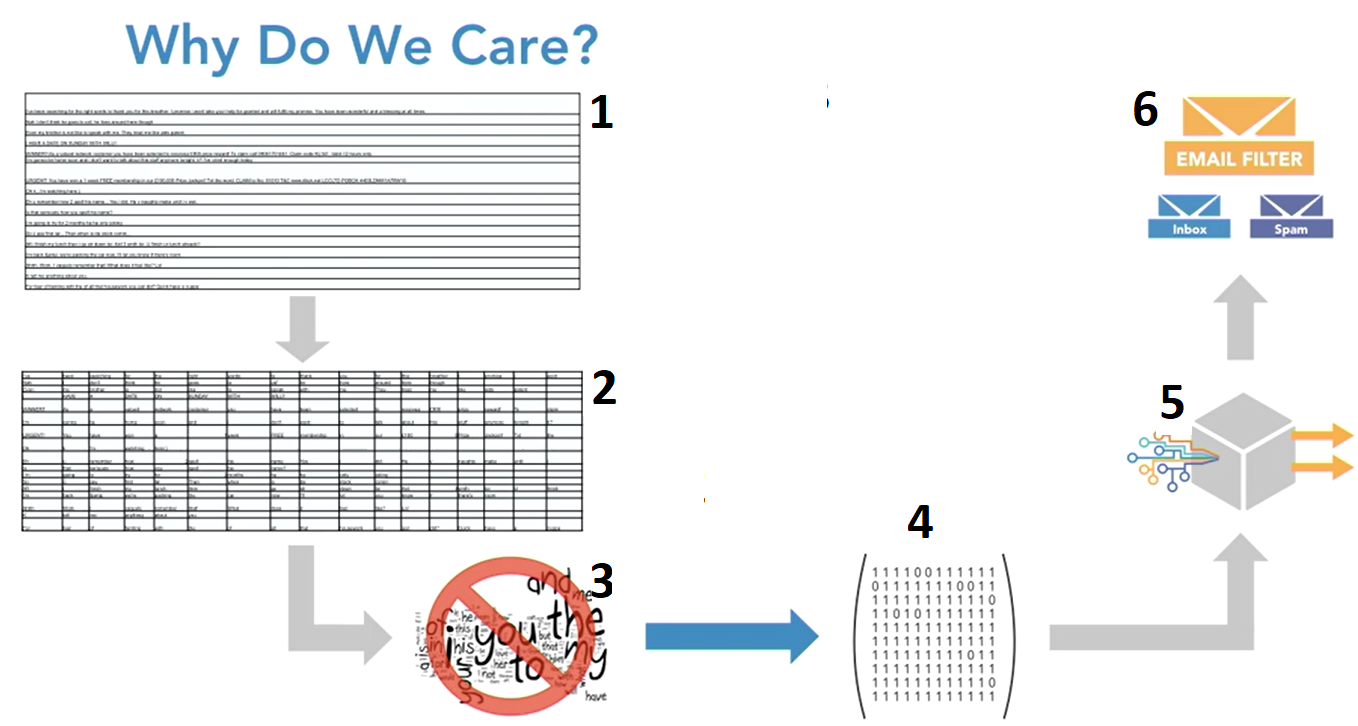
*Start a new Python project folder … do not reuse the previous workspace … As with any keyboard-driven console-like environment, developing muscle -memory for the common commands is also part of the learning curve.*

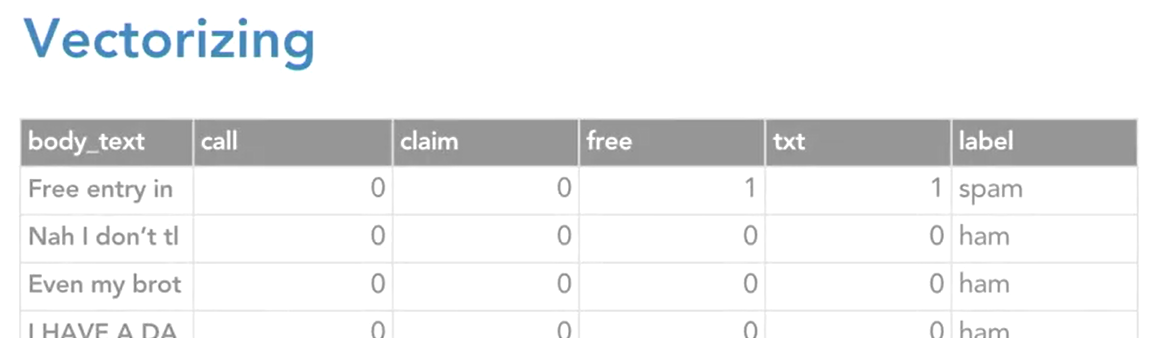
***Vectorizing*** is the process to convert text into a form that Python and a machine learning model can understand. It encode text as integers to create feature vectors.

A ***feature vector*** is an n-dimensional vector of numerical features that represent some object.

In our context, that means we’ll be taking an individual text message and converting it to a numeric vector that represents that text message. The diagram below is a summary of our machine learning pipeline.



We are now in step #4 where we vectorize text. Where we will take the dataset that had one line per document, then we will convert it into a matrix that still has one line per document, but then you have every word used across all documents as the columns of your matrix.

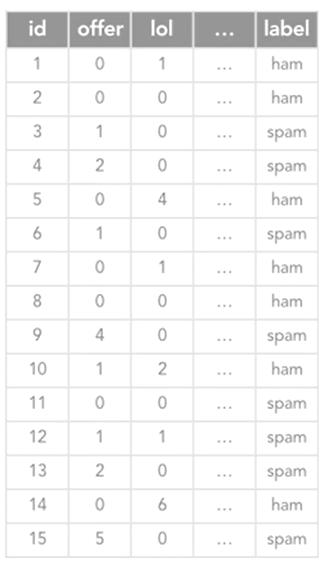


And then within each cell is counting how many times that certain word appeared in that document. The matrix above is called the document term matrix. It contains the numeric representation of each text message, then we can carry on down the pipeline and fit and train a model.

Why Do We Care?

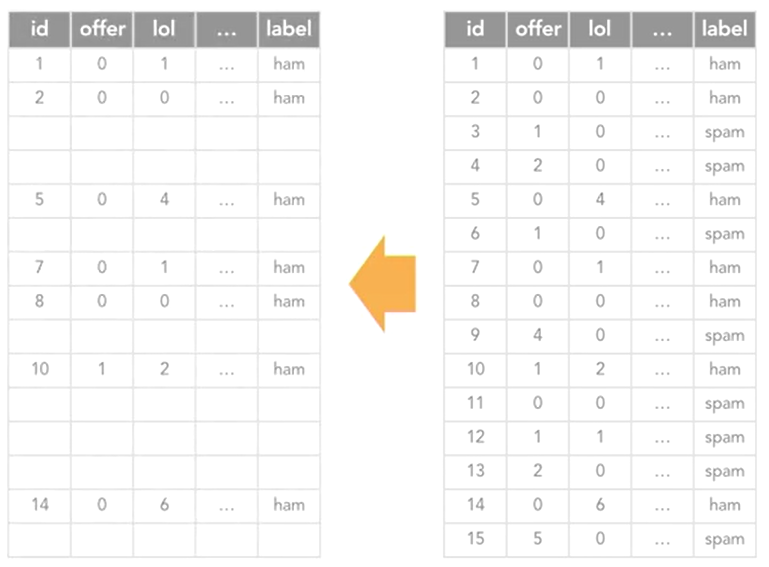
* When looking at a word, Python only sees a string of characters.
* Raw text needs to be converted to numbers so that Python and the algorithms used for machine learning can understand

Vectorization Example

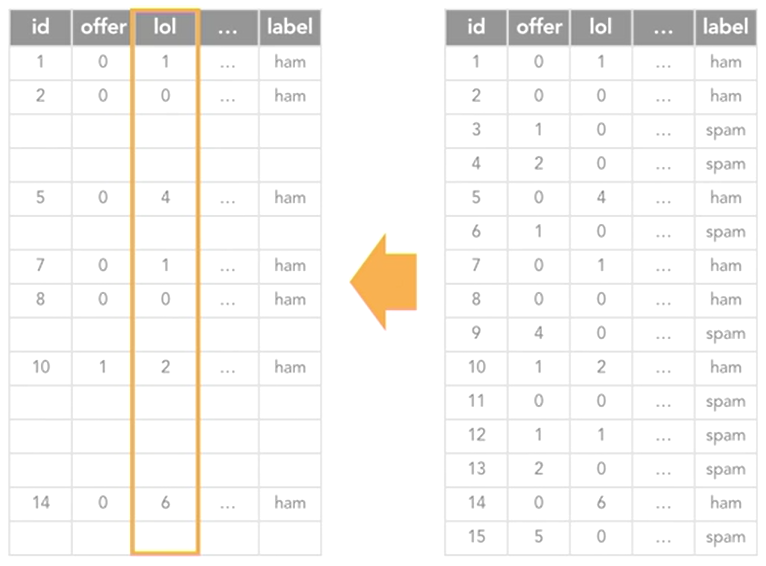


Here, we only have 15 text messages and we want to classify them each as either spam or ham. Let’s just say there are 400 unique words across those 15 text messages. That would mean our matrix, after vectorizing, would still have 15 rows just like this one does, one per text message. And it would have 400 columns, one for each unique word. And each cell entry would count the number of times that word was used in each text message. Again, this is called your document term matrix. This matrix here is a subset of the full document term matrix. We’re just focusing on two words here used in text messages, offer and lol, along with the label of either spam or ham.

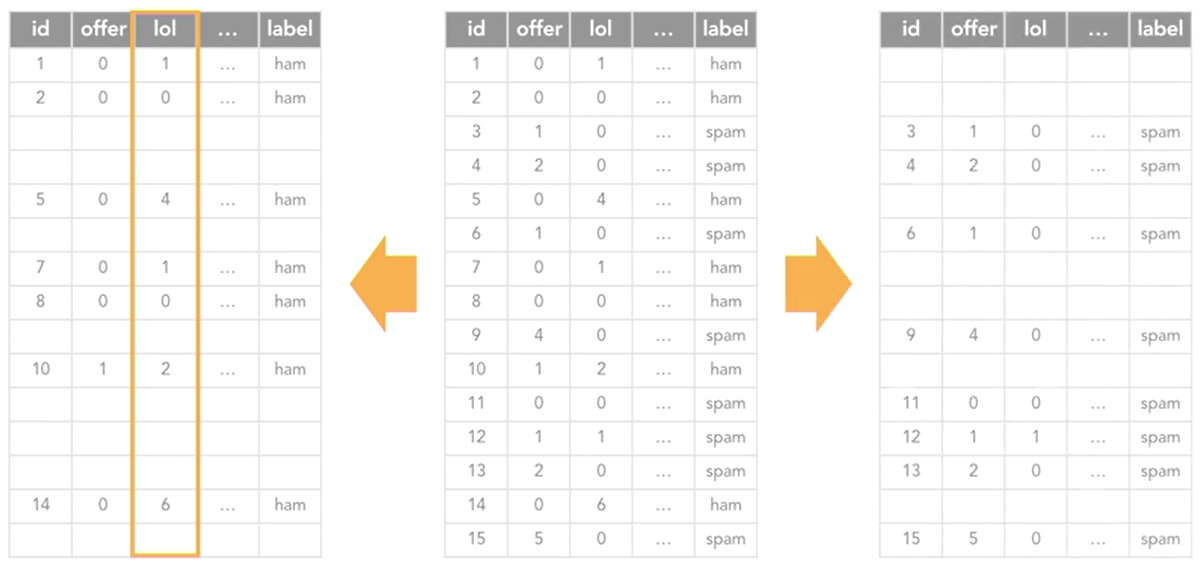
But again, in reality, this … here would represent the other 398 words that were used across these text messages. So that’s what this looks like after vectorizing. Now how does a machine learning model use this information to learn what these words mean? I mentioned before that by looking at the counts in the cells, that it can start to correlate which words happen in combination with certain labels. So let’s isolate just the non-spam messages here. You can start to notice that offer occurs very frequently, but lol occurs in a lot of these non-spam text messages.



So in the first message, offer occurs zero times, lol occurs once, and you can imagine that that would also occur with many other words that we’re not showing here. In the second text message, neither occur. And then in the fifth, offer doesn’t occur at all, lol occurs four times, and son on. So from the numbers here, the model could pretty easily pick up on the fact that lol occurs quite frequently with non-spam text messages and offer occurs very infrequently. You could see how this would allow a model to start to learn how to predict when a text is spam or not, based just on the text body.



So now, let’s do the same thing, but we’ll jump over to the spam messages. You can pretty quickly notice that it’s the opposite. Offer occurs quite frequently, while lol occurs quite infrequently. So the model would pick up on the fact that offer occurs quite frequently in spam messages and lol occurs quite infrequently. So now, only consider these two words, the model has learned that offer occurs frequently with spam and infrequently with non-spam, while lol occurs frequently with non-spam and infrequently with spam.



So you could see how maybe with even just these two words, the model could start making ham or spam predictions about new text messages based only on the number of times these two words occur. With that said, this is an extremely simple and exaggerated example for the purpose of illustration. In reality, the model would need to learn the relationships of much more than two words to make an accurate prediction. But this example was meant to show how vectorizing helps the model roughly learn what words correlate with which labels.

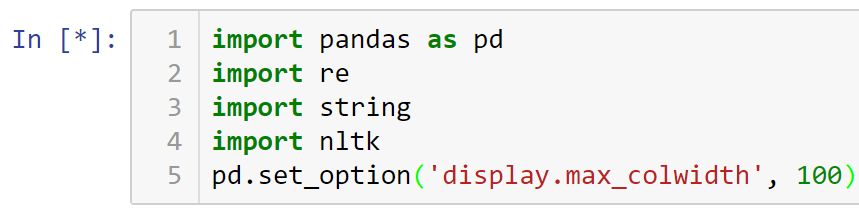
Different Types of Vectorization

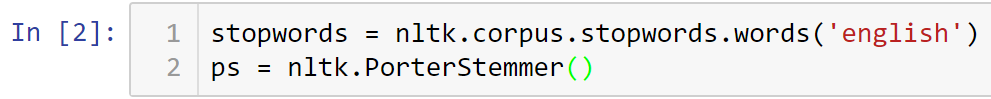
* Count vectorization
* N-grams
* Term frequency - inverse document frequency (TF-IDF)

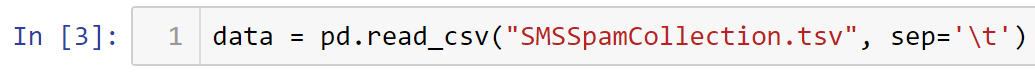
So far I’ve entirely been talking about the entry of each cell in the document term matrix containing the count of how many times a given word appears in that text message. But that’s only one method of vectorization. And that’s called, not surprisingly, count vectorization. There are two other variations of count vectorization called N-grams and term frequency - inverse document frequency, which is often referred to as TF-IDF.

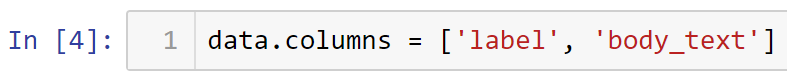
Implementing Count Vectorization

Count vectorization creates a document-term matrix where the entry of each cell will be a count of the number of times that word occurred in that document, or text message in our case, and that’s what’s stored in the given cell, so it’s pretty straight forward.

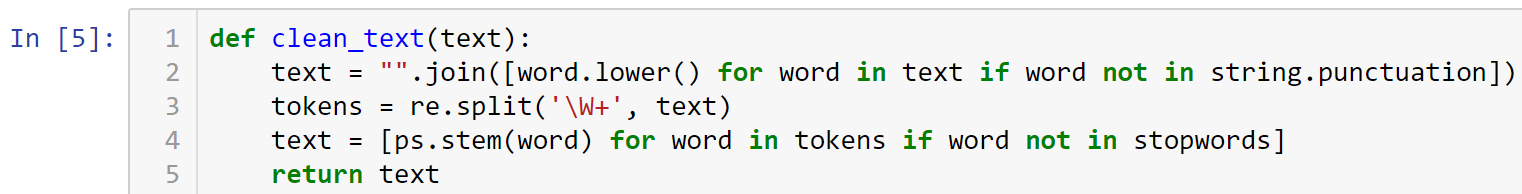




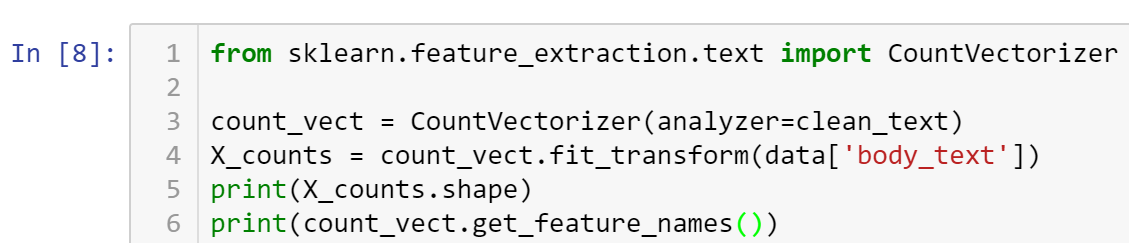


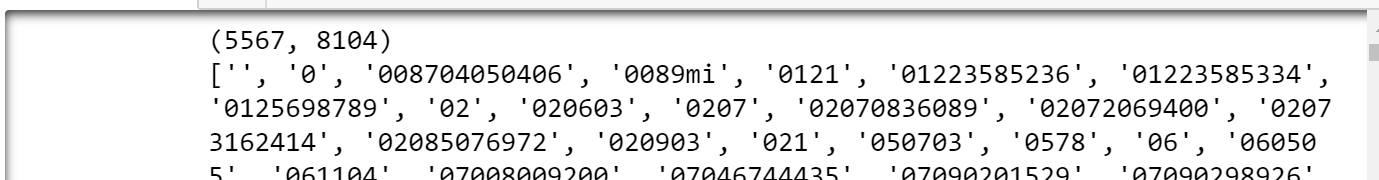


Creating function to remove punctuation, tokenize, remove stopwords, and stem

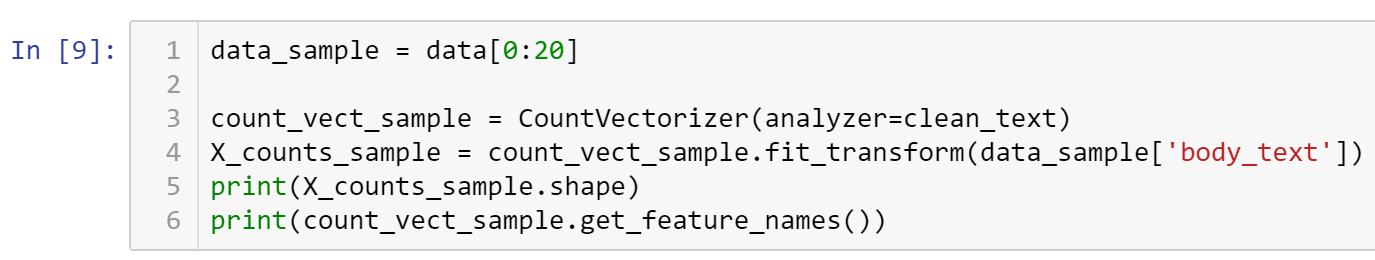


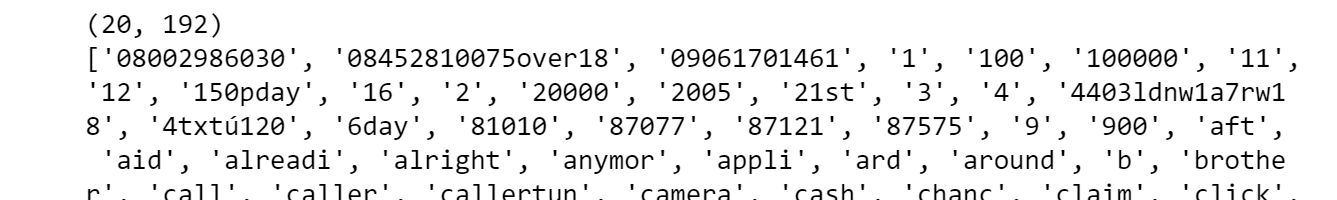
Apply CountVectorizer





Apply CountVectorizer to smaller sample



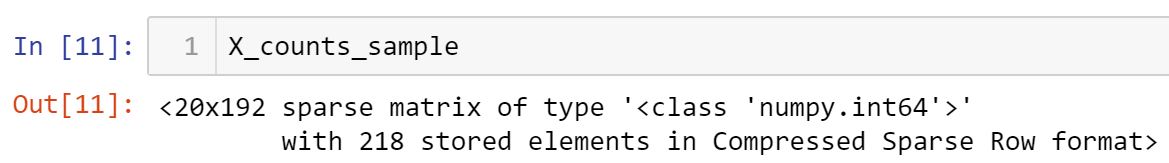


The new matrix sample above is more digestible, made up of 20 rows and 192 columns.

The raw output of a vectorizer function is known as sparse matrix.

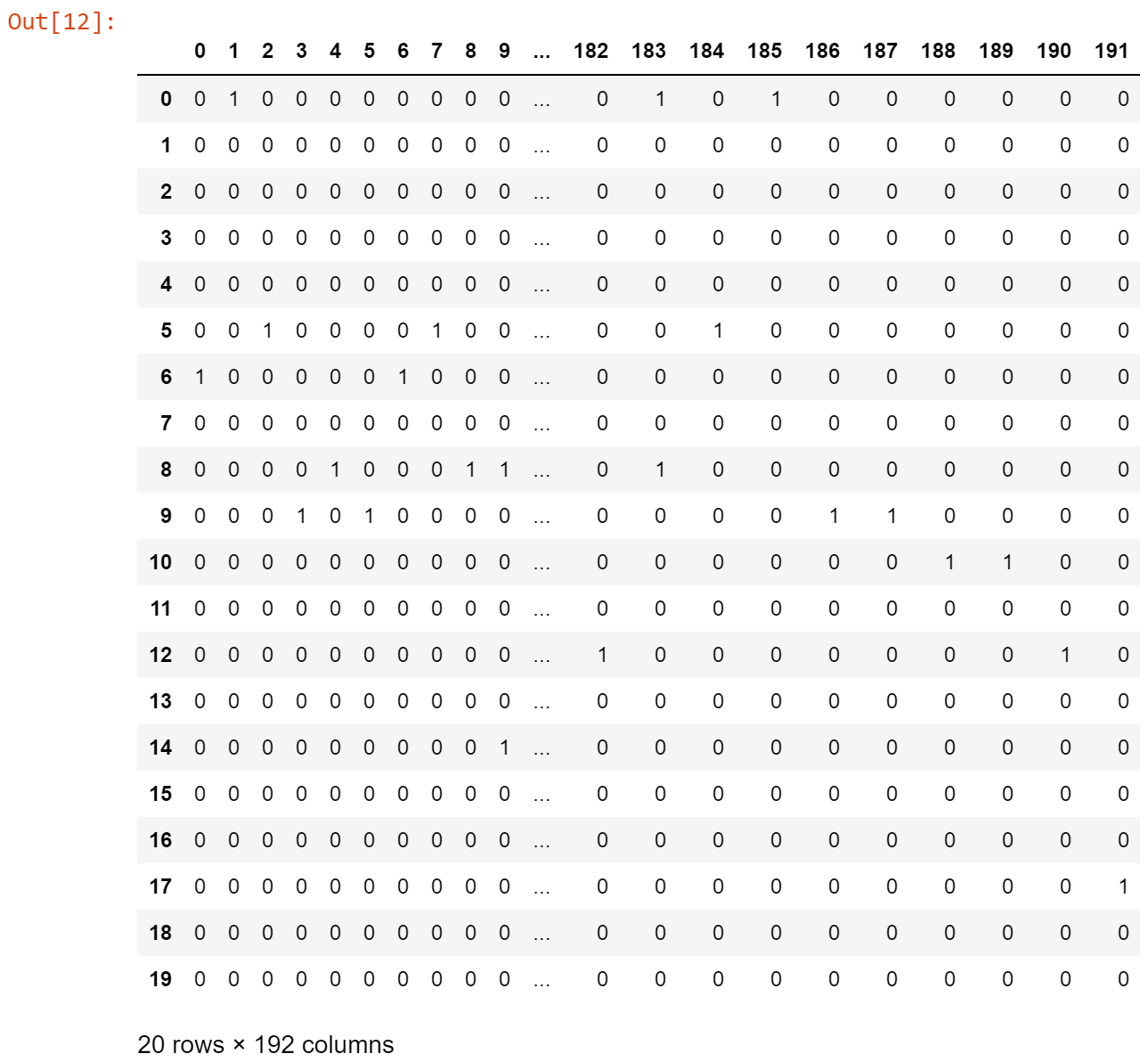
Sparse Matrix: A matrix in which most entries are 0. In the interest of efficient storage, a sparse matrix will be stored by only storing the locations of the non-zero elements.

Displaying x\_counts\_sample (with capital X) will yield the property of the matrix.

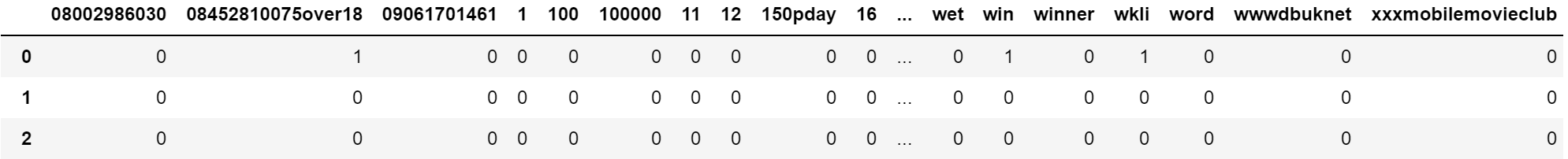
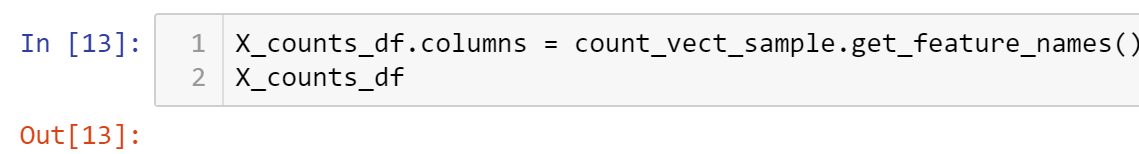


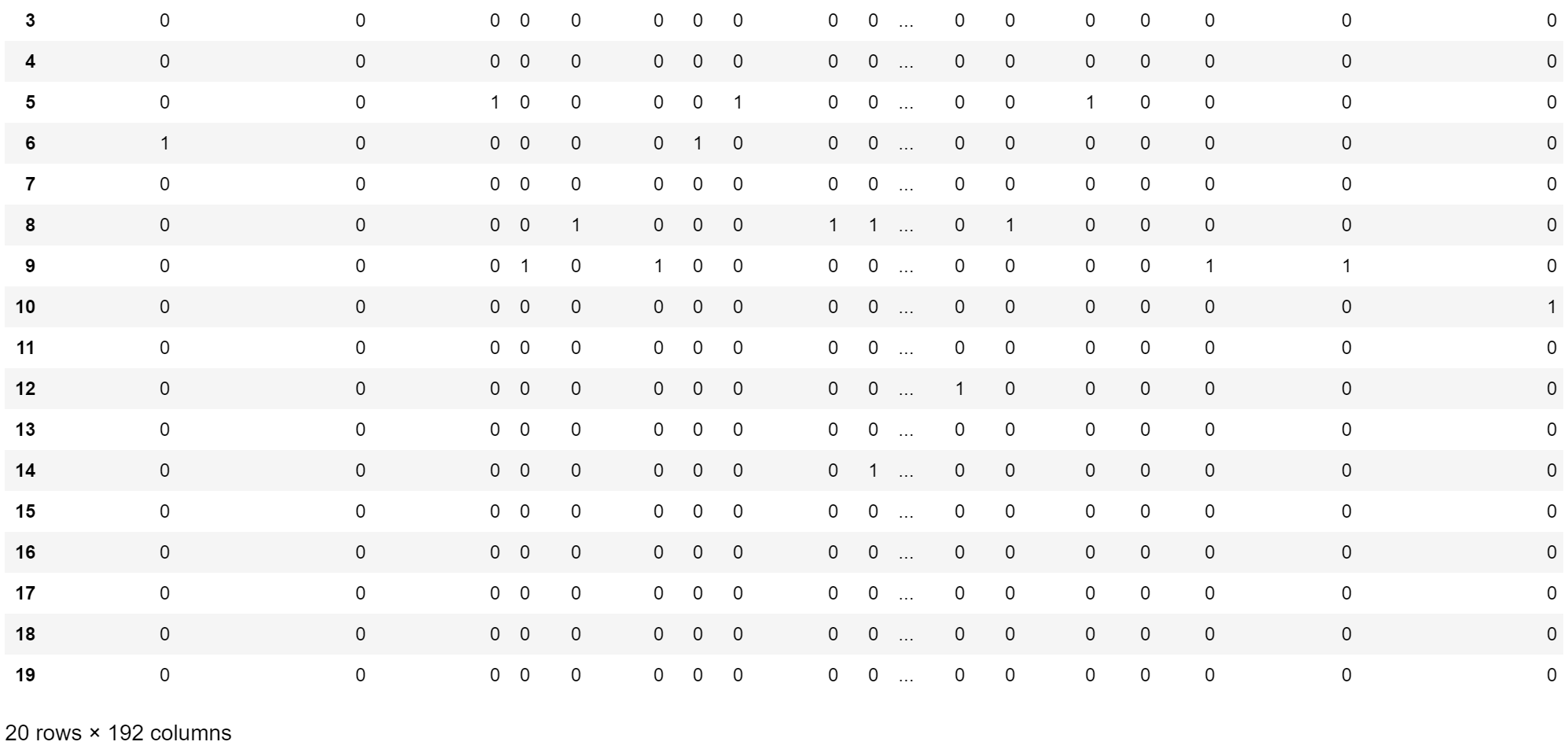
To display the actual contents, you need to wrap that in a DataFrame …





Note that the column names are the the word text, but numbers. Python will make the association of the words to the numeric equivalent. Let’s convert those column numbers to the words they represent.





Now the column labels are displayed.

* All submissions should be separate from other exercises and quests. Please do not lump all your answers into one document and re-using that same workspace to gain multiple points. Thanks.
* Place your name at the bottom of your code, download your Python program in html format, and submit your work in Canvas.